Politechnika Śląska Biblioteka Główna
P.1877/89

6 1989

informatyka

Prof. W. Koczkodaj
o arkuszach kalkulacyjnych

Superkomputery CRAY

WYDAWNICTWO NOT SIGMA

Cena 300 zł

informatyka

6 1989

Nr 6
Miesięcznik Rok XXIV
Czerwiec 1989

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:
Mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA
(sekretarz redakcji),
Władysław KLEPACZ
(redaktor naczelny),
dr inż. Marek MACHURA,
dr inż. Wiktor RZECZKOWSKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA,
dr inż. Janusz ZALEWSKI
(zastępca redaktora naczelnego).

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18 m. 17, tel. 39-14-34

RSW „PRASA-KSIĄŻKA-RUCH"
PRASOWE ZAKŁADY GRAFICZNE
ul. Dworcowa 13, 85-950 BYDGOSZCZ
Zam. 4626/88.
Obj. 4,0 ark. druk. Nakład 8500 egz.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 300 zł
Prenumerata roczna 3600 zł

WYDAWNICTWO NACZELNA ORGANIZACJA TECHNICZNA
SIGMA
CZASOPISM I KSIĄŻEK TECHNICZNYCH

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE: Strona

Arkusz kalkulacyjny jako kolejny krok w kierunku nieprogramowania
Waldemar W. Koczkodaj 1

Nowa generacja superkomputerów Cray (1). Cray X–MP
Zdzisław Szczerbiński 6

Przyspieszanie obliczeń przez ich rozpraszanie (2). Rozpraszanie programu
Wacław Iszkowski 9

Nowa norma Fortranu (2)
Jacek Skrzymowski 12

Grafika w Turbo Pascalu 4.0
Jan Bielecki 15

ZE ŚWIATA

Problemy stosowania Ady
Turbo Pascal 4.0
Edukacja informatyczna w Japonii
18. Konferencja GfI

NOWE KSIĄŻKI

Arkusze obliczeniowe 4

Bazy danych 30

TERMINOLOGIA III okł.

Słownik pojęć i terminów z dziedziny rozpoznawania i przetwarzania obrazów

W NAJBLIŻSZYCH NUMERACH:

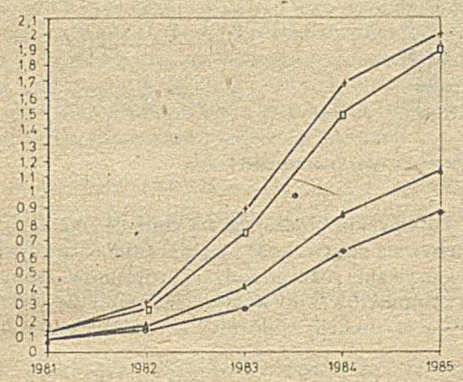
- Felix Redmill opisuje zalecenia Komitetu Technicznego nr 7 międzynarodowej organizacji EWICS, dotyczące bezpieczeństwa i niezawodności systemów komputerowych
- Janusz Zalewski analizuje pierwszy zbiór zaleceń Komitetu Technicznego nr 7, dotyczących wytwarzania i użycia systemów komputerowych w zastosowaniach krytycznych
- Marek Gondzio, Marek Pawłowski i Jacek Wytrębowicz omawiają narzędzia ułatwiające i przyspieszające uruchamianie systemów mikroprocesorowych
- Jacek Kuraś, Jacek Lembas i Marek Skomorowski charakteryzują język GODYS–PC, przeznaczony do symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi, na mikrokomputerach typu IBM PC XT/AT
- Marzena Tataj i Alina Toruń zajmują się zastosowaniem tomografii komputerowej w medycynie
- Jacek Irlik przedstawia stanowisko PTI w sprawie ochrony prawnej programów komputerowych

WALDEMAR W. KOCZKODAJ
Department of Mathematics
and Computer Science
Laurentian University
Sudbury, prowincja Ontario
Kanada

POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach
P. 1877/89

Arkusz kalkulacyjny jako kolejny krok w kierunku nieprogramowania

Mija już dziesięć lat od chwili opracowania pierwszego arkusza kalkulacyjnego VisiCalc przez Daniela Bricklina i Roberta Franstona (obaj studiowali administrację na Uniwersytecie Harwardzkim). Sukces systemu VisiCalc był natychmiastowy i zaskakujący nawet dla jego twórców – sprzedaż 100 000 egzemplarzy w pierwszym roku była niebywałym osiągnięciem przyczyniającym się, według wielu ekspertów, do rewolucji informatycznej. Do sukcesu pierwszego popularnego komputera osobistego Apple przyczynił się w dużej mierze właśnie VisiCalc, ponieważ niektóre firmy kupowały ten komputer jedynie w celu użycia arkusza kalkulacyjnego do sporządzania kosztorysów i bardzo pracochłonnych analiz finansowych. VisiCalc został zaimplementowany niemalże na wszystkich popularnych komputerach osobistych, stając się jednocześnie wzorcem do naśladowania dla przyszłych arkuszy kalkulacyjnych, które obecnie są absolutnie nieodzowne do prowadzenia każdego biznesu. Popularność arkuszy ilustruje wykres przedstawiony na rysunku.



Tendencja rozwoju programów aplikacyjnych (w tysiącach sztuk rocznie; kolejno – arkusze elektroniczne, procesory tekstu, bazy danych i programy komunikacyjne)

LOTUS 1–2–3

Najbardziej reprezentatywnym arkuszem kalkulacyjnym jest Lotus 1–2–3. W chwili obecnej zdominował on zupełnie rynek amerykański i zawędrował niemal pod każdą strzechę w Ameryce Północnej. Jest standardem przemysłowym i każdy inny arkusz kalkulacyjny musi być kompatybilny (w sensie logiki lub akceptowania plików) z Lotusem, aby mieć jakiekolwiek szanse na rynku. W tym artykule używam Lotusa do zilustrowania podstawowych pojęć związanych z arkuszami kalkulacyjnymi.

Uniwersalność stosowania arkusza kalkulacyjnego do rozwiązywania nawet skomplikowanych problemów, włącznie z równaniami różniczkowymi cząstkowymi, jest związana z niekonwencjonalnym podejściem do programowania. Pomimo niebywałego sukcesu w biznesie, arkusz jest zaskakująco nieznany w kręgach naukowych i komputerowych. „Prawdziwemu" programiście arkusz kalkulacyjny za bardzo kojarzy się ze znienawidzonymi księgami finansowymi. Tak więc, nie powinno czytelnika zdziwić, iż moc obliczeniowa arkusza może być bardziej doceniona przez pracownika – powiedzmy – Gminnej Spółdzielni niż pracownika nauki. Zilustruję to prostym przykładem spisu towarów (por. tabela).

Przykładowy spis towarów

Numer komórki	A	B	C	D
1	Towar	Ilość	Cena	Wartość
2	Gwoździe	20	60	1200
3	Deski	120	200	24000
4	Cement	50	750	37500
5	Pustaki	5000	90	450000
6	Razem			512700

Typowy spis towarów zawiera kilkaset elementów przy cenie często zmieniającej się w krótkim okresie (w Ameryce Płn. może to zachodzić nawet codziennie). Kolumny *A*, *B* i *C* z tabeli są wprowadzone jako dane wejściowe. Kolumna *D* zawiera wyniki powstałe przez pomnożenie zawartości kolumn *B* i *C*. Liczby te oczywiście zmienią się, gdy ilość lub cena ulega zmianie. Zawartość komórki *D2* jest obliczana według wzoru: + *B2*C2*. Umieszczenie znaku + na początku wzoru powoduje, że ten napis jest traktowany jako wzór a nie etykieta (tekst). Służy to również do graficznego konstruowania wzorów i stanowi jedną z najmocniejszych stron Lotusa, szczególnie ważną dla osób nie zaznajomionych z pojęciami programowania.

Aby więc wprowadzić wzór do komórki *D2* umieszcza się w niej wskaźnik (podświetlenie) używając strzałek klawiatury numerycznej. Następnie wpisuje się znak + na miejsce kursora umieszczonego w drugim wierszu monitora i przesuwa się wskaźnik do komórki *B2*, co powoduje automatyczne pojawienie się napisu *B2* ze znakiem plus. Następnie należy dopisać znak mnożenia * (w tym momencie wskaźnik automatycznie cofa się do komórki *D2*) i przesunąć wskaźnik do komórki *C2* co powoduje automatyczne dopisanie napisu *C2* do wzoru na miejsce kursora. Oczywiście użytkownik ma możliwość rozbudowania wzoru opisanym sposobem lub przez dopisywanie innych elementów (w szczególności można użyć np. dzielenia przez 2, aby uzyskać połowę wartości) maksymalnie do 256 znaków. Nietrudno się domyśleć, że podobny wzór powinien być wprowadzony do komórki *D3*, tzn. + *B3*C3*.

Graficzne konstruowanie wzoru, choć bardzo proste, jest dość czasochłonne. W wypadku kilkuset elementów mogłoby zniechęcić nawet najwytrwalszego użytkownika, a na pewno nie znalazłoby dużego uznania np. wśród biznesmenów na Wall Street. Otóż najbardziej użytecznym poleceniem w każdym niemalże arkuszu kalkulacyjnym (porównywalnym do instrukcji przypisania w języku programowania wysokiego poziomu) jest polecenie kopiowania (*COPY*). Dostęp do trybu poleceń jest w arkuszu zapewniony zazwyczaj przez pewien wyszczególniony klawisz. W Lotusie odpowiada mu znak dzielenia „/". Tak więc, po przesunięciu wskaźnika do komórki *D2* naciska się klawisz „/" i wybiera *C*, aby rozpocząć operację kopiowania (*COPY*). Lotus podpowiada *D2* (komórka, w której jest umieszczony wskaźnik) jako komórkę kopiowania (*FROM*). W tym momencie można oczywiście wskaźnik komórki przesunąć w inne miejsce (przy użyciu klawiszy strzałek) lub zaakceptować przez naciśnięcie klawisza *ENTER* (co wykonujemy w naszym przykładzie). Następnie przesuwamy wskaźnik do komórki *D4* i kotwiczymy go przez naciśnięcie klawisza kropki (na ekranie ukazuje się *D3*.. *D3* jako podpowiedź zakresu docelowego).

W naszym przykładzie chcemy skopiować wzór do więcej niż jednej komórki w jednym podejściu. W tym celu przesuwamy wskaźnik do komórki *D5* (dwa razy dolną strzałką) i naciskamy klawisz *ENTER*. W drugim wierszu na monitorze pojawia się podpowiedź *D3*.. D5. Ku zdziwieniu sceptyków rezultatem naszej operacji kopiowania są poprawne liczby, nie zaś 1200 powtórzone 5 razy. Operacja */C* kopiuje zawartość komórki, którą w przykładzie jest wzór, nie zaś liczba 1200. Liczba 1200 jest (czasowym) wynikiem obliczenia według tego wzoru, nie zaś zawartością komórki *D2*. „Cudowna" zmiana liczby 2 na 3, 4 i 5 we wzorze z komórki *D2* jest związana z pojęciem rodzaju użytecznej referencji.

Identyfikacja „szachowa" komórki przez literę (lub dwie litery dla kolumn wykraczających poza dwudziestą szóstą) i cyfry (lub liczby z przedziału [1;8192]) jest przykładem referencji względnej, która ulega zmianie przy zastosowaniu operacji kopiowania. Kopiowanie według wierszy powoduje zwiększenie numeru wiersza o jeden. Kopiowanie według kolumn spowodowałoby „zwiększenie" litery. Na przykład, skopiowanie komórki *D2* do *F2* zmieniłoby wzór $+B2*C2$ na $+D2*E2$, dając oczywiście wynik 0, gdyż *E2* jest komórką pustą. Komórka *D6* zawiera wzór ⓐ*SUM (D2.. D6)* tworzony również w sposób graficzny. Tworzenie tej zawartości rozpoczynamy od umieszczenia wskaźnika komórki w *D6* i wpisania w pierwszej części wzoru ⓐ SUM (–zakończonej lewym nawiasem okrągłym. Zakres działania funkcji sumowania (wszystkie funkcje Lotusa rozpoczynają się od znaku ⓐ) można wpisać lub zdefiniować graficznie przez przesuwanie wskaźnika i zakotwiczenie kropką.

W tym miejscu, czytelnik z wykształceniem informatycznym dojdzie do oczywistego wniosku, że napisanie równoważnego programu w Basicu nie powinno mu zabrać więcej niż 10–15 min. Gdzie jest więc ta oszczędność? Po pierwsze, nie jest możliwe nauczenie Basica, w zakresie potrzebnym do napisania powyższego programu, losowo wybranego pracownika, np. Gminnej Spółdzielni w tym samym czasie, który wystarcza do zrozumienia, jak skonstruować wyżej opisany arkusz kalkulacyjny. Po drugie raz skonstruowany arkusz kalkulacyjny może być modyfikowany nieskończoną liczbę razy, przy naprawdę niewielkim nakładzie pracy. Na przykład, zmiana liczby desek ze 120 na 180 (przez wpisanie tej wartości do komórki *B3*) spowoduje natychmiastowe przeliczenie nie tylko wartości składowanych desek, ale również całkowitej wartości towarów. Jest to tak zwana analiza *„co się stanie, jeżeli..."* (ang. *what-if analysis*). W wypadku Basica należałoby zmienić instrukcję *DATA* (co wbrew pozorom, dla laika nie musi być banalne w wypadku powiedzmy 500 wierszy *DATA*), wykonać program, odszukać i zinterpretować wyniki.

PRZYKŁADY PRAKTYCZNE

ANALIZĘ *„co się stanie, jeżeli"* zilustruję na przykładzie, jak zostać milionerem (dolarowym). Załóżmy, że zamierzamy produkować buty, towar szczególnie „chodliwy" (również w dosłownym znaczeniu) w okresie zimowym.

C5: (C0) [W12] +C3*C4

	A	B	C	D
1	Przewidywania produkcyjne			
2				
3	Cena jednostkowa		$20	
4	Wielkosc sprzedazy		1200	
5	Wartosc sprzedazy		$24,000	
6				
7	Koszty			
8	produkcji		$10,000	
9	distrybucji		$1,400	
10	rynkowe		$5,000	
11	Narzuty		$3,000	
12				
13	Koszt calkowity		$19,400	
14				
15	Zysk netto		$4,600	
16				
17				
18				
19				
20				

15-Jan-89 03:02 PM

Wydruk 1. Jak się dorobić na produkcji butów

Dane do przykładu przedstawiono na wydruku 1. Wzory są zawarte w następujących komórkach:

C5: $+C3*C4$
C13: ⓐ *SUM (C8..C11)*
C15: $+C5-C13$

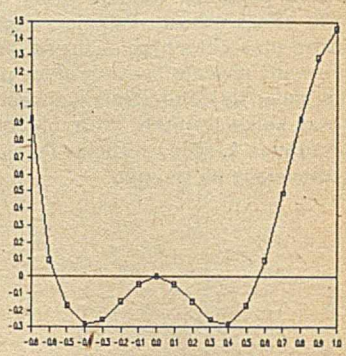
Szerokość kolumny *C* została zmieniona na 12 znaków (przez polecenie */WCS12*), aby wyświetlić większe liczby w formacie walutowym (uzyskanym przez polecenie */RFC*).

Ustawmy wskaźnik komórki w *C3*. Na początek zmienimy cenę butów na 500 dolarów (prawdopodobnie będą to złote ciżemki). Zysk netto 580 600 dolarów (ponad pół miliona) wydaje się być dobrą zachętą do dalszych eksperymentów. Sprawdźmy, kiedy dorobimy się miliona (czemu by nie). Uzyskuje się to w prosty sposób przez zmianę wartości w komórce *C3*, powiedzmy, na 900, 800 dolarów, co daje kolejne liczby: 1 080 000 i 960 000 dolarów. Po kilku następnych próbach (według zasad podziału binarnego, w mniej niż $\log_2 50$, tzn. w sześciu krokach) dojdziemy do wniosku, że cena 850 dolarów daje nam upragniony milion. Po minucie zastanowienia dojdziemy jedak do wniosku, że przy cenie 850 dolarów za parę trudno będzie znaleźć 1200 nabywców, nawet jeżeli będziemy sprzedawali złote ciżemki (szczególnie zimą). Sprawdźmy więc, jak przedstawia się sytuacja po sprzedaniu bardziej realistycznej ilości 3 par (przez podstawienie 3 do komórki *C4*), pozostawiając ciągle tę samą wartość 850 dolarów w *C3*. Nie powinniśmy być zdziwieni, że wynik jest zupełnie zniechęcający – 16 850 dolarów (w nawiasie, dla liczb ujemnych – według konwencji używanej w księgowości). Tak się zazwyczaj kończy próba szybkiego dorobienia się, czego niestety większość niedoszłych biznesmenów doświadcza raczej na własnej skórze niż na arkuszu kalkulacyjnym.

B3: [W9] 0.5*X^2-X*@SIN(5*X)

	A	B	C
1		x	f(x)
2			
3		0.3	-0.24424
4		0.5	-0.17423
5		0.6	0.095327
6		0.7	0.490548
7		0.8	0.925441
8		0.9	1.284777
9		1	1.458924
10		1.1	1.381094
11		1.2	1.065298
12		1.3	0.565344
13		1.4	0.060218
14		1.5	-0.28199
15		1.6	-0.30297
16		1.7	0.087571
17		1.8	0.878186
18		1.9	1.947787
19		2.0	3.088042
20		2.1	4.052361

15-Jan-89 03:37 PM



Wydruk 2. Obliczanie i wykreślanie wartości funkcji

Operacja kopiowania polegająca na zmianie referencji względnych zapewnia niebywałą elastyczność i adaptawalność arkusza kalkulacyjnego do rozwiązywania różnego rodzaju problemów. Jednym z częstych zastosowań komputerów w doświadczeniach naukowych i technicznych jest tabelaryzacja funkcji. Załóżmy, że należy wyświetlić wartości funkcji

$$y = 0,5x^2 - x \cdot \sin 5x$$

w przedziale [–0,3; 2,1] z krokiem 0,1. Budowę arkusza powinniśmy oprzeć na wzorze rekurencyjnym

$$x := x + krok$$

Implementacja tego podstawowego skądinąd wzoru sprowadza się do chytrego użycia operacji kopiowania. Mianowicie, w komórce *A3* umieszczamy wartość początkową –1, zaś w *A4* wzór $+A3+0.1$ (dodaj 0,1 do poprzedniej wartości, aby otrzymać wartość bieżącą). Komórkę *B3* wypełniamy wzorem

$$0.5*x\uparrow 2 - x*ⓐ sin(5*x)$$

po uprzednim nadaniu komórce *A3* nazwy *x* poleceniem */RNC*. Teraz można skopiować komórkę *B3* do *B4*, a następnie zakres *A4..B4* do 17 kolejnych komórek poczynając od *A5*. Czytelnik nie powinien martwić się, gdy skopiuje ten zakres do większej liczby komórek niż 17. Pozostałe komórki można zupełnie zignorować lub usunąć z arkusza poleceniem */RE*.

Należy zauważyć, iż po zmianie funkcji (np. współczynnik 0,5 można zastąpić przez 0,73) należy skopiować komórkę *B3* do pozostałych komórek w kolumnie *B* przez ustawienie wskaźnika komórki na *B3* i operację

/C [ENTER] [DOLNA-STRZAŁKA] . [END] [DOLNA-STRZAŁKA] [ENTER]

Sekwencja tych ośmiu znaków może być zdefiniowana jako makrowywołanie, które później wykonuje się przez naciśnięcie klawisza *ALT* z literą użytą w nazwie tego makrowywołania. Podobnie dzieje się ze zmianą kroku, który jest użyty we wzorach. Zmiana wartości początkowej (w naszym wypadku –1) odbywa się jedynie przez umieszczenie innej liczby w *A3*, gdyż jest to jedyne miejsce, gdzie wartość ta została umieszczona. Pozostałe komórki zawierają jedynie odsyłacze do tej wartości.

Uzyskane wartości funkcji można wykreślić przy minimalnym nakładzie pracy przez użycie polecenia */G*. Należy zdefiniować dwa zakresy: *A3..A23* dla zmiennej niezależnej (*X-range*) i *B3..B23* dla zmiennej zależnej (*A-range*). Funkcję wyświetla się przez użycie klawisza *V* (ang. *view* – podgląd) w trybie grafu lub przez naciśnięcie klawisza funkcyjnego *F10* w normalnym trybie wprowadzania danych. Efekt jest dość szokujący, gdyż osiągnięcie podobnego rezultatu na dużym komputerze jest poza zasięgiem możliwości przeciętnego użytkownika (ze względu na konieczność dostępu do biblioteki graficznej i plotera) albo zbyt pracochłonne. Pierwszy tego rodzaju eksperyment kończy się zazwyczaj wynikiem w postaci paru bezsensownych kresek, „wyjazdem" piórka poza obręb zamierzanego rysunku i paroma pejoratywnymi komentarzami ze strony pań operatorek na temat umiejętności programisty. Oczywiście, zastosowanie programu kreślącego typu graph-in-the-box na komputerze osobistym może przynieść nawet lepsze efekty wizualne, jednakże kosztem ręcznego wprowadzenia danych, ponieważ programy graficzne zazwyczaj nie pozwalają na łatwiejsze generowanie wartości funkcji.

ODWRACANIE MACIERZY

Prawdopodobnie najbardziej zaskakujące rezultaty otrzymuje się przy odwracaniu macierzy. Po pierwsze, wprowadzenie macierzy jest nadzwyczaj łatwe – następuje automatyczne wyrównywanie w kolumnach i natychmiastowe wyświetlanie wprowadzonych danych. Po drugie, możliwe jest zastosowanie pewnych reguł sprawdzania poprawności danych, np. czy suma elementów równa się jeden (częsta sytuacja dla problemów związanych z rachunkiem prawdopodobieństwa), znalezienie wartości maksymalnej (funkcja @*MAX*), minimalnej (@*MIN*) lub średniej @*AVG*). Najważniejsza jest jednak prostota operacji sprowadzającej się do następującej sekwencji poleceń:

/Data Matrix Invert [*END*] [*PRAWA-STRZAŁKA*] . [*END*] [*DOLNA-STRZAŁKA*] [*ENTER*] [*END*] [*DOLNA-STRZAŁKA*] [*DOLNA-STRZAŁKA*] [*ENTER*]

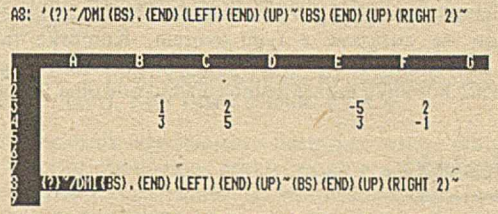
gdzie w nawiasach kwadratowych podano nazwy odpowiednich klawiszy.

Dla niedoświadczonego użytkownika powyższa sekwencja może okazać się dość skomplikowana. Jest to jednak jedynie czternaście przyciśnięć, które są podpowiadane przez program. Na przykład, po naciśnięciu znaku „/" (tryb polecenia) i *D (Data)* w drugim wierszu ekranu ukazuje się menu. Według konwencji Lotusa można wskazać nazwę polecenia kursorem lub nacisnąć klawisz pierwszej litery tej nazwy (w naszym wypadku *M*). Kombinacja [*END*] [*STRZAŁKA*] powoduje przesunięcie wskaźnika komórki na koniec kolumny lub wiersza. W praktyce, wysyłanie powyższej sekwencji poleceń zajmuje przeciętnemu użytkownikowi zaledwie kilka sekund.

Jak wiadomo, Lotus pozwala też na zdefiniowanie makrowywołania, które jest później wykonywane przez jednoczesne naciśnięcie klawisza *ALT* i jego nazwy (a jest nią pojedyncza litera). Jak łatwo zauważyć, w ten sposób można zdefiniować jedynie 26 makrowywołań. Po pierwsze, jednak, jest to i tak o wiele więcej niż nasza możliwość pamiętania, a po drugie, można zdefiniować makrowywołanie odczytujące praktycznie każdą nazwę i wykonujące procedurę o tej nazwie. Następujące makrowywołanie umożliwia wprowadzenie macierzy i jej odwrócenie:

{?}~ */DMI* {*BS*} {*END*} {*LEFT*} {*END*} {*UP*}~ {*BS*} {*END*} {*UP*} {*RIGHT 2*}~

Powyższe makrowywołanie jest kompletne i powinno być wpisane w całości do jednej komórki, którą koniecznie należy nazwać ciągiem poleceń */RNC* (ang. *Range Name Create*) używając jako nazwy ukośnika „\" (ang. *backslash*) i jednej litery (powiedzmy *M*) – por. wydruk 3. Wszystkie znaki wchodzą w skład definicji, a znaczenie poszczególnych symboli jest następujące:
{?} – wprowadzić dane, ile i gdzie potrzeba; koniec operacji następuje po dwukrotnym naciśnięciu klawisza *ENTER*,
/ – tryb polecenia
DMI – polecenia *Data Matrix Invert*,
{*END*} – klawisz *END*,
{*LEFT*} – klawisz „strzałka w lewo",
{*RIGHT*} – klawisz „strzałka w prawo",
{*UP*} – klawisz „strzałka do góry",
{*BS*} – znak usuwania (ang. backspace),
~ – wężyk (tylda), oznacza klawisz *ENTER*,
{*RIGHT 2*} – skrócona forma równoważna dwukrotnemu użyciu klawisza {*RIGHT*} {*RIGHT*}



Wydruk 3. Definicja makrowywołania do wprowadzenia i odwrócenia macierzy

Definicja makrowywołania zawiera pełną gamę niezbędnych poleceń z ulubioną przez informatyków rekurencją włącznie. Na przykład, umieszczenie {*M*} na końcu definicji opisanego makrowywołania spowoduje jego wykonanie przed zakończeniem. W Lotusie mamy więc fundamentalną instrukcję przypisania {*LET ...*}, instrukcję warunkową {*IF ...*}, instrukcję skoku, instrukcję procedury {\ *nazwa*} i pętli {*FOR ...*}. Dodatkowo, Lotus zawiera wszelkie funkcje matematyczne, finansowe, tekstowe, statystyczne i specjalne. Jedne z najciekawszych funkcji, @ *CELLPOINTER* i @, umożliwiają badanie komórek i traktowanie ich zawartości jako odsyłaczy do innych komórek. Tworzenie więc list rekurencyjnych i odsyłaczy do odsyłaczy jest bardzo proste.

ZAKOŃCZENIE

W tym miejscu chciałbym zrobić dygresję do filozofii programowania. W Polsce, przy programowaniu pokutuje zasada „zadziwić świat cały" (przestawienie wyrazów jest celowe), tj. tendencja wytwarzania programów doskonale uniwersalnych i „nieprzemakalnych". Podejście to (związane ze sposobem kształcenia informatyków, czemu warto poświęcić oddzielny artykuł) nie jest niestety wbrew pozorom rozsądne, gdyż produkcja takich programów zajmuje dużo czasu i jest zbyt kosztowna. Tak wyprodukowane programy zajmują również więcej pamięci i są wolniejsze w działaniu niż oparte na zdroworozsądkowych zasadach. Lotus jest uważany powszechnie za arcydzieło inżynierii programowania. Podstawowym elementem sukcesu Lotusa jest jego „praktyczność" przejawiająca się w zapewnieniu maksymalnej kontekstowej pomocy, dobrze zaprojektowanych wartościach początkowych i domyślnych, w graficznym ukierunkowaniu systemu i w ignorowaniu wszelkich nierealnych założeń, takich jak zbyt duża swoboda wyboru czy nienaturalna długość wyrażeń (ciekaw jestem, na przykład, ilu czytelników „*Informatyki*" zakodowało „ciurkiem" instrukcję dłuższą niż 256 znaków?).

Czytelnik z wykształceniem informatycznym zada sobie prawdopodobnie jedno pytanie: dlaczego arkusz kalkulacyjny spowodował rewolucję w koncepcjach programowania, a przynajmniej wprowadził sporo zamieszania? Aby na to odpowiedzieć, należy cofnąć się dość daleko do początków historii programowania. Jak wiadomo, pierwsze komputery miały na celu wyeliminowanie zbyt wolnych mechanicznych części ruchomych. Do przetwarzania liczb zostały użyte układy lampowe. Każdy programista (a raczej, monter-elektryk) wiedział dokładnie, gdzie (fizycznie) znajdowała się powiedzmy suma liczb, bo nierzadko programowanie sprowadzało się do zmiany (przelutowania) przewodów.

W miarę rozwoju programowania, odstępowano od konieczności pamiętania miejsc fizycznych użytych do przechowywania wartości. Od powstania języków asemblerowych powszechnie używano symboli do oznaczania obiektów obliczeniowych (niegdyś popularna nazwa „języki symboliczne" wskazuje na docenienie tego wielkiego kroku naprzód).

Ten proces abstrakcji doprowadził do „triumfu nauki nad zdrowym rozsądkiem" (ulubione powiedzenie mojego profesora od mechaniki teoretycznej). W szczególności, w językach programowania wysokiego poziomu (np. Basicu) tworzy się zmienne, o niewiadomym znaczeniu, które nie sposób zlokalizować w komputerze (zwłaszcza z pamięcią wirtualną). Pojęcie zmiennej jest niewątpliwie jedynym z najtrudniejszych pojęć w programowaniu. Jest to również (wbrew pozorom) bariera nie do przebycia dla osób z odmiennym od matematycznego postrzeganiem rzeczywistości. Nic więc dziwnego, że arkusz kalkulacyjny całkowicie zrewolucjonizował podejście do programowania zastępując zmienną („ściśle tajny" obiekt typu UFO) przez namacalną komórkę! Na pytanie, gdzie znajduje się suma, każdy użytkownik arkusza kalkulacyjnego wskaże miejsce na ekranie z okrzykiem na ustach: *„o tutaj!"*. W Basicu należy użyć instrukcji *PRINT* w programie (który należy wykonać), aby ujrzeć tę wartość. W Lotusie nie trzeba robić nic. Wartość zmiennej (a zbrodnią jest utożsamiać ją ze zmienną) jest wyświetlana natychmiast po wprowadzeniu wzorów lub zmianie innych wartości, które są związane z tym wzorem. Komórka ma swój „szachowy" identyfikator, który można interpretować jako adres fizyczny. Każdej komórce można jednak nadać dowolną liczbę nazw.

Co więcej można zrobić, aby umożliwić „ludzkie" (przez co rozumiem nieskomplikowane czyli nienaukowe) używanie komputera? Jest to niewątpliwie potężny krok naprzód w stronę rozsądnego użycia komputerów nie mający, nawiasem mówiąc, nic wspólnego ze sztuczną (a złośliwi twierdzą, że z żadną) inteligencją. Wydaje się jednak, że gdzieś tam po drodze (naukowej) zatracono zdrowy rozsądek. Potrzeba było 35 lat i dwóch laików komputerowych (studentów administracji) do odkrycia tego, co było przez informatyków skrzętnie utajniane: zmiennej. Być może jest to głównym powodem powszechnego i ostentacyjnego ignorowania arkusza kalkulacyjnego w świecie (a raczej, światku pseudo-) naukowym, pomimo już dziesiątej rocznicy powstania tego użytecznego systemu.

Arkusz kalkulacyjny jest narzędziem zupełnie niezastąpionym w początkowym nauczaniu programowania. Dla tych, którzy mi nie wierzą, mam proste pytanie: *„Dlaczego w biologii naukę rozpoczyna się od pierwotniaków (organizmów jednokomórkowych), a nie powiedzmy systemu moczowego wielbłądów?"*. Galileusz (gdyby dożył naszych burzliwych czasów) prawdopodobnie skwitowałby sprawę stwierdzeniem: *„A jednak komórka!"*.

Nowe książki

Arkusze obliczeniowe

A.T. Williams: What if? A User's Guide to Spreadsheets on the IBM PC. Wiley and Sons, Chichester (UK), 1984. ISBN 0471-89218-1

Książka stanowi wprowadzenie do elektronicznych arkuszy obliczeniowych (ang. *spreadsheets*) i wyjaśnia, do czego służą, jak działają i jak mogą być używane. Zawiera zwarty, porównawczy opis wszystkich popularnych arkuszy dla komputerów IBM PC, na poziomie nie wymagającym żadnej uprzedniej wiedzy informatycznej. Praktyczne informacje, jak najlepiej wykorzystać program i jak ominąć jego ograniczenia, są ujęte w specjalnych ramkach. Ponadto, książka zawiera ponad 100 tabel, rysunków i zdjęć.

Spis treści. Część I – skuteczne tworzenie elektronicznych arkuszy obliczeniowych. Wprowadzenie. Tworzenie arkuszy. Podstawy analizy *„co się stanie, jeżeli"*. Zapis arkusza na dysk. Drukowanie arkuszy. Funkcje logiczne i tablice wyszukiwawcze. Część II – ocena elektronicznych arkuszy obliczeniowych. Visicalc, Lotus 1-2-3, Supercalc 2, Multiplan, Context MBA.

W.J. Doyle: Using Supercalc – The Next Generation. Wiley and Sons, Chichester (UK), 1986. ISBN 0471-80828-8

Jest to wprowadzenie do zasad działania i stosowania elektronicznych arkuszy obliczeniowych, przy użyciu programów Supercalc, Supercalc 2 i Supercalc 3. Jako jedyna książka na rynku zawiera syntetyczne omówienie wszystkich trzech wersji arkusza. Opisano w niej składnię wszystkich poleceń i funkcji, ze szczególnym uwzględnieniem największych możliwości programu Supercalc 3. Opisy poparto wieloma przykładami i zaleceniami, jak tworzyć i rozmieszczać plansze do zastosowań w zagadnieniach zarządzania.

Spis treści. Wprowadzenie (Czym jest elektroniczny arkusz obliczeniowy? Środowisko programu Supercalc 3). Podstawowe cechy programu Supercalc (Zasady pracy z programem. Wprowadzenie danych. Globalne sterowanie arkuszem. Sterowanie wyświetlaniem. Operowanie arkuszem. Wprowadzanie i wyprowadzanie arkuszy. Funkcje). Nowoczesne właściwości programu Supercalc 3 (Dokładniejsze sterowanie wyświetlaniem arkuszy. Wyszukiwanie i sortowanie. Zintegrowana grafika. Zarządzanie danymi. Użycie plików rozkazowych. Sprzęganie z innymi pakietami). Zastosowanie do obliczeń w zarządzaniu.

J. Willis, W. Pasewark: Supercalc 3 – Learning, Using and Mastering. Tab Books, Blue Ridge Summit (PA), 1986. ISBN 8-8306-26948

Książka jest przeznaczona dla czytelników nie mających do czynienia z arkuszami elektronicznymi, pragnących nauczyć się posługiwania programem Supercalc i ogólnie – komputerami w zagadnieniach zarządzania. Będzie także przydatna dla obecnych użytkowników tego programu zainteresowanych przejściem do stosowania wersji 3 oraz – dla zawodowych programistów wykorzystujących dotychczas inne arkusze obliczeniowe. W książce zawarto wiele danych użytkowych i przykładowych programów, ilustrującychużycie Supercalac od najbardziej podstawowych procedur do najbardziej złożonych funkcji. Nowsze cechy, jak grafika i sortowanie, przedstawiono w sposób pogłębiony, na poziomie profesjonalnym, co umożliwia zawodowym programistom pełne wykorzystanie zaawansowanych właściwości programu na takich komuterach, jak IBM PC lub Apple II. Znaczną zaletą tej książki jest dodatek zawierający zwięzłe omówienie wszystkich poleceń, funkcji i procedur programu, o wiele dokładniejsze niż w podręczniku użytkowania lub w innych materiałach firmowych dotyczących Supercalca. Dobra organizacja książki, poparta licznymi przykładami i wyjaśnieniami, umożliwia jej efektywne wykorzystanie w praktycznych zagadnieniach zarządzania.

E. H. Lee: The SuperCalc SuperBook. The Offical Guide to SuperCalc 2, SuperCalc3, and SuperCalc 4. Sigma Press, Wilmslow (UK), 1988. ISBN 1850580804

Książka dowodzi, że arkusze elektroniczne mogą służyć nie tylko jako pomoc w prowadzeniu interesów i w analizach finansowych, lecz także w wielu innych dziedzinach działalności ludzkiej. Niedocenianie znaczenia arkuszy przez zwykłych użytkowników komputerów prowadzi do niewykorzystania dużej części ich mocy obliczeniowej. Autor stara się uświadomić ten fakt możliwie największej liczbie użytkowników, przez obszerne wprowadzenie do przedmiotu i dobór przykładów. W dalszej części książki omówiono i wyjaśniono najtrudniejsze fragmenty podręcznika firmowego, podając wiele szczegółów technicznych, na ogół niedostępnych skądinąd. Przede wszystkim przedstawiono cechy doliczeniowe arkuszy, omawiając także ich użycie jako baz danych, pakietów statystycznych, do wykreślania rysunków i pisania makropoleceń. Podano też wiele szczegółów nt. drukowania i uyżcia sprzętu.

D. Bocolan: Lotus 1-2-3 Simplified – Including Version 2.0. Tab Books, Blue Ridge Summit (PA), 1986. ISBN 0-8306-27480

Jest to drugie wydanie tej książki, poprawione, uaktualnione i rozszerzone o wersję 2.0 programu Lotus 1-2-3, mającą znacznie większe możliwości funkcjonalne i szybszą niż poprzednia, a ponadto zgodną pod względem systemu plików i makrowywołań z pakietem Symphony, i pozbawioną ograniczenia rozmiaru programu do 640 KB, które miała wersja poprzednia. Autor przedstawił tę tematykę przy użyciu licznych przykładów praktycznych, ilustrując je na ponad 20 arkuszach kalkulacyjnych. Uważna lektura przykładów umożliwia opanowanie umiejętności samodzielnego tworzenia arkuszy kalkulacyjnych do analiz finansowych, gospodarki magazynowej, prowadzenia księgowości, tworzenia listy płac itd. Oprócz tradycyjnej tematyki, w książce omówiono też nowsze zagadnienia, jak np. ochronę danych za pomocą haseł, przyspieszenie obliczeń przez użycie koprocesora, programowanie przy użyciu makrowywołań i in.

Spis treści. Co może Lotus 1-2-3? Wymagania systemowe, wyposażenie dodatkowe i instalowanie. Klawiatura i system operacyjny. Zasady

użytkowania programu. Operacje plikowe. Podstawowe polecenia. Wygląd arkusza. Drukowanie arkuszy. Operowanie dużymi arkuszami. Tworzenie wykresów. Drukowanie wykresów. Zarządzanie bazą danych. Polecenia zaawansowane. Programowanie makrowywołań. Tworzenie tzw. kart czasowych. Zagadnienia księgowania.

A.T. Williams: Lotus 1-2-3. A Self Teaching Guide. Wiley and Sons, Chichester (UK), 1986. ISBN 0471-84461-6

Jest to kompletny podręcznik omawiający wszystkie zaawansowane właściwości wersji 2.0 programu Lotus 1-2-3, przeznaczony dla doświadczonych użytkowników. Wyjaśniono w nim m.in., jak używać makrowywołań, list menu tworzonych przez użytkownika, konwersji typów danych, funkcji napisowych oraz sprzęgania z systemami plików pierwotnej wersji 1-2-3 i programów Jazz i Symphony. Wszystkie polecenia opisano za pomocą tzw. struktur drzewiastych i szczegółowych instrukcji. Użycie poszczególnych funkcji zilustrowano przykładami, zaopatrując je w uwagi dotyczące najbardziej efektywnego wykorzystania. Podano również ostrzeżenia i wskazówki dotyczące postępowania w sytuacjach awaryjnych.

Spis treści. Współpraca z drugą wersją programu Lotus 1-2-3. Polecenia dotyczące arkuszy. Polecenia odnoszące się do zakresów. Polecenia kopiowania, przenoszenia i zakończenia pracy. Polecenia dotyczące plików. Polecenie drukowania. Tworzenie arkuszy. Polecenia bazy danych. Polecenia dodatkowe. Polecenia tworzenia wykresów. Makrowywołania – ukryta siła programu 1-2-3.

M. Jackson: Advanced Spreadsheet Modelling with Lotus 1-2-3. Wiley and Sons, Chichester (UK), 1988. ISBN 0471-91989-6

Ciekawą innowacją wprowadzoną w wydaniu tej książki jest towarzysząca jej dyskietka zawierająca pliki danych dla opisywanych przykładów. Treść książki ilustruje tezę, że zakres zastosowań Lotusa jest znacznie szerszy niż dziedzina analiz finansowych, gdzie program jest najchętniej używany. Książka jest przeznaczona dla menedżerów stosujących w praktyce arkusze kalkulacyjne, i znających już Lotusa. Oprócz typowych zagadnień omówiono w niej prognozowanie, podejmowanie decyzji, planowanie, kontrolowanie wydatków budżetowych i symulację. Na tym tle przedstawiono, jak można używać programu Lotus 1-2-3 w charakterze osobistego narzędzia do modelowania i tworzyć arkusze kalkulacyjne służące jako modele powyższych zastosowań.

Spis treści. Wprowadzenie. Modele oceniające. Wykresy do przedstawienia danych i modelowania. Modele finansowe – planowanie. Modele finansowe – analiza budżetu. Prognozowanie w prowadzeniu interesów. Modele decyzyjne. Podstawy modelowania w makroskali. Modele symulacyjne.

C. Gilbert, L. Williams: The ABC's of 1-2-3 – Release 3. Sybex, Alameda (CA), 1988. ISBN 0-89588-519

Książka jest świetnym wprowadzeniem do trzeciej wersji programu Lotus 1-2-3, przeznaczonym dla początkujących użytkowników i osób zainteresowanych stosowaniem komputerów w pracy zawodowej. Omówiono w niej najbardziej podstawowe czynności praktyczne, zaczynając od włączenia komputera i instalowania programu 1-2-3, przechodząc stopniowo do tworzenia i używania arkuszy kalkulacyjnych, by skończyć na tworzeniu wykresów, zarządzaniu bazą danych i analizie *„co się stanie, jeżeli..."*. Napisana na poziomie elementarnym książka będzie z pewnością przydatna wszystkim pragnącym zrozumieć sens stosowania komputerów w przesiębiorstwach.

G. Harvey, K.Y. Nelson: Lotus 1-2-2 Instant Reference. Second Edition. Sybex, Alameda (CA), 1988. ISBN 0-89588-553

Ta książka jest przeznaczona dla aktualnych użytkowników trzeciej wersji programu Lotus 1-2-3. Stanowi bardzo przydatny w pracy zawodowej przewodnik dostarczający szybkiej informacji na temat poszczególnych poleceń, ich opcji, funkcji itp. Odnosi się zarówno do systemu PC-DOS, jak i OS-2. Zawiera też opis sposobów łączenia arkuszy kalkulacyjnych, jak i omówienie nowych, pełniejszych możliwości graficznych. Jako źródło szybkich informacji może być przydatna zawodowym programistom, lecz także wszystkim osobom nie mającym jeszcze usystematyzowanych wiadomości na temat Lotusa.

G. Weisskopf: Lotus 1-2-3 Tips and Tricks. Sybex, Alameda (CA), 1988. ISBN 0-89588-454-2

Książka jest nieocenionym zbiorem sztuczek i chwytów programistycznych, przeznaczonym dla najbardziej doświadczonych użytkowników Lotusa. Zawarte w niej informacje odnoszą się tylko do drugiej wersji programu i dotyczą m.in. makrowywołań, tworzenia zakresów, projektowania arkuszy, nietypowego korzystania ze sprzętu, programowania na poziomie systemu operacyjnego, drukowania, wymiany danych, opracowania programów aplikacyjnych itp. Jest nieocenionym źródłem informacji dla twórców poważniejszych zastosowań.

J. R. Ottensman: Quattro Simplified. TAB Books, Blue Ridge Summit (PA), 1988. ISBN 0-8306-3111-9

Ta książka, napisana językiem łatwo czytelnym dla menedżerów, zawiera wszystkie wiadomości niezbędne do szybkiego zapoznania się z użytkowaniem arkusza Quattro i przeprowadzenia przykładowych, lecz realistycznych obliczeń. W poszczególnych rozdziałach omówiono kolejno: tworzenie i modyfikowanie arkuszy, zarządzanie plikami, operowanie wykresami, zarządzanie bazą danych, podstawowe funkcje matematyczne i finansowe, użycie bardziej złożonych funkcjiw analizie czułości, analizie statystycznej i operacjach macierzowych, użycie makrowywołań, włącznie z wywołaniami typowymi dla pakietu Lotus 1–2–3 w Quattro.

E. R. Tymes, C. E. Prael: Quattro for the Professional. TAB Books, Blue Ridge Summit (PA), 1988 ISBN 0-8306-9378-5

Jest to podręcznik wyjaśniający i ilustrujący użycię arkusza obliczeniowego Quattro. Za pomocą praktycznych przykładów i klarownego wykładu książka uczy czytelnika, jak wprowadzać dane, operować nimi i analizować je bezpośrednio w arkuszu, bądź w bazie danych lub w postaci graficznej. Omówiono szczegółowo budowę poleceń, opierając się na analogii z Lotusem 1–2–3 i podkreślając ich większą przyjazność i łatwość uyżcia. Wyjaśniono dostosywalność pakietu do poszczególnych zastosowań i odmiennych wymagań użytkowych. W bardziej złożonych przykładach przedstawiono opis rzeczywistych problemów występujących w prowadzeniu interesów. Książka będzie szczególnie użyteczna dla osób, które już wcześniej używały w praktyce arkuszy elektronicznych, a obecnie chciałyby zwiększyć możliwości posługując się programem Quattro.

A. Simpson, D. J. Wolt: Mastering Quattro. Sybex, alameda (CA), 1989. ISBN 0-89588-514-X

Celem książki jest takie zapoznanie czytelników z arkuszem obliczeniowym Quattro, aby opanowali go maksymalnie szybko, nieczego niezaniedbując z bogatej wiedzy o tym pakiecie. Materieł podstawowy obejmuje tworzenie arkuszy i wprowadzanie danych, a materiał trudniejszy – analizę danych, operacje macierzowe, histogramy oraz użycie makrowywołań i menu użytkownika.

JANUSZ ZALEWSKI

Instytut Podstaw Informatyki PAN w Warszawie organizuje
w dniach 21 – 24 maja 1990 r. w Kazimierzu Dolnym
I Konferencję naukową pn.

GRAFIKA KOMPUTEROWA PRZETWARZANIE OBRAZÓW

Tematyka konferencji: języki i bazowe systemy graficzne, bazy danych graficznych, algorytmy dla grafiki i przetwarzania obrazów; nowe rozwiązania sprzętowe, wybrane zastosowania itp.

W programie przewidziano kilka wykładów przez zaproszonych specjalistów oraz prezentację zgłoszonych prac. Termin nadsyłania referatów upływa 30 października 1989 roku.

Bliższe dane na temat konferencji można uzyskać u jej organizatorów pod adresem:
Konferencja GKPO '90, IPI PAN
ul. Polna 18/20,
00–625 Warszawa,
tel. 25-18-97

ZDZISŁAW SZCZERBIŃSKI
Zakład Systemów Automatyki Kompleksowej
Polska Akademia Nauk
Gliwice

Nowa generacja superkomputerów Cray (1)

Cray X-MP

W ostatnich latach obserwuje się udane próby łączenia dwóch metod przetwarzania równoległego: przetwarzania potokowego (ang. *pipeline processing*) i przetwarzania wieloprocesorowego (ang. *multiprocessing*). Taką drogę obrała, przodująca w świecie w dziedzinie produkcji superszybkich komputerów (superkomputerów), firma Cray Research, wprowadzając na rynek modele Cray X-MP i Cray-2. Stanowią one wieloprocesorowe rozwinięcie potokowego komputera Cray-1, przedstawionego już w Informatyce nr 5, 1985. W poniższym artykule omówiono najważniejsze zagadnienia architektury i oprogramowania maszyn Cray X-MP, a w drugiej części zaprezentowano superkomputer CRAY-2.

ARCHITEKTURA

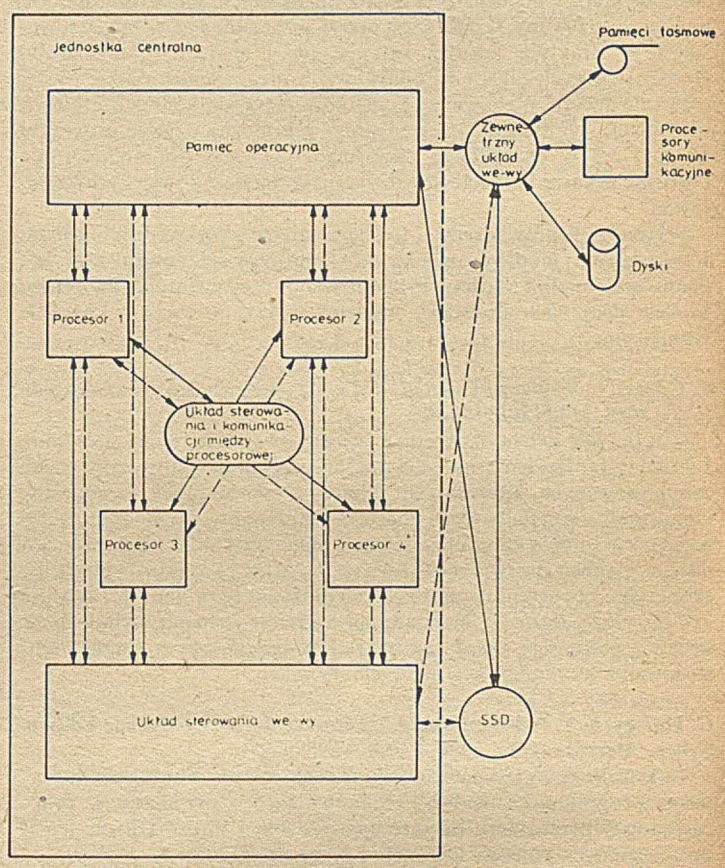
Architektura superkomputera Cray X-MP jest dziełem byłego głównego projektanta systemów w firmie Cray Research, Steve'a Chena. Stanowi ona udoskonalenie projektu Cray-1, którego autorem był założyciel firmy, Seymour Cray. Podstawową innowacją jest zwielokrotnienie udoskonalonego wektorowego procesora, zastosowanego w maszynie Cray-1, z wykorzystaniem wspólnej pamięci operacyjnej. Istnieją również modele jednoprocesorowe Cray X-MP, szybsze od Cray-1. Dwa pierwsze, dwuprocesorowe modele maszyny Cray X-MP zostały wprowadzone na rynek w 1982 r. W 1984 r. pojawiły się cztery nowe modele i w tym momencie zaprzestano produkcji systemu Cray-1. Od 1985 roku pojawiają się nowe wersje, charakteryzujące się zazwyczaj nieco mniejszą mocą obliczeniową i niższą ceną. W tabeli 1 zaprezentowano dostępne obecnie konfiguracje systemu Cray X-MP, z uwzględnieniem różnic w liczbie procesorów, rozmiarze pamięci operacyjnej i liczbie jej modułów.

Tabela 1. Konfiguracje superkomputera Cray X-MP

Model	Liczba procesorów	Rozmiar pamięci (mln słów)	Liczba modułów pamięci
X-MP/11	1	1	16
X-MP/12	1	2	16
X-MP/14	1	4	16
X-MP/18	1	8	32
X-MP/24	2	4	16
X-MP/28	2	8	32
X-MP/216	2	16	32
X-MP/48	4	8	32
X-MP/416	4	16	64

Na rysunku 1 przedstawiono ogólny schemat blokowy systemu Cray X-MP w maksymalnej, 4-procesorowej konfiguracji. Jednostka centralna łączy się z komputerami komunikacyjnymi oraz z urządzeniami pamięci masowej (dyski, pamięci taśmowe) za pośrednictwem zewnętrznego układu we-wy. Zewnętrzna pamięć półprzewodnikowa SSD (ang. *Solid-State Storage Device*) stanowi odpowiednik pamięci podręcznej (ang. *cache*) między pamięciami: masową i operacyjną. Procesory komunikują się ze sobą za pośrednictwem specjalizowanego układu, który równocześnie zapewnia synchronizację ich pracy.

Wewnętrzną strukturę procesora wektorowego, użytego w systemie X-MP, przedstawiono na rys. 2. Jest ona niemal identyczna ze strukturą procesora Cray-1. Wyróżnić w niej można pięć sekcji. Sekcja wektora realizuje obliczenia wykonywane na wektorach danych, w sposób potokowy. Składa się na nią szereg jednostek funkcjonalnych (arytmetyczno-logicznych), zestaw rejestrów wektorowych oraz rejestry maski i długości wektora. Sekcja skalarna wykonuje obliczenia na pojedynczych danych (skalarach). Oprócz jednostek funkcjonalnych, w jej skład wchodzą rejestry skalarne (przechowujące dane i wyniki obliczeń dokonywanych w jednostkach funkcjonalnych) oraz bufor między rejestrami skalarnymi i pamięcią. Sekcja adresowa jest odpowiedzialna za efektywne adresowanie pamięci. Zawiera własne jednostki funkcjonalne, realizujące proste (stałoprzecinkowe) operacje arytmetyczne na adresach i indeksach, zestawy rejestrów adresowych oraz bufor adresów. Sekcja rozkazów pobiera i dekoduje rozkazy z pamięci. Obejmuje ona bufory paczek rozkazów i rejestry rozkazów. Sekcja we-wy ma własny port do pamięci i umożliwia operacje we-wy niezależnie od przesłań między pamięcią a sekcją wektorową.

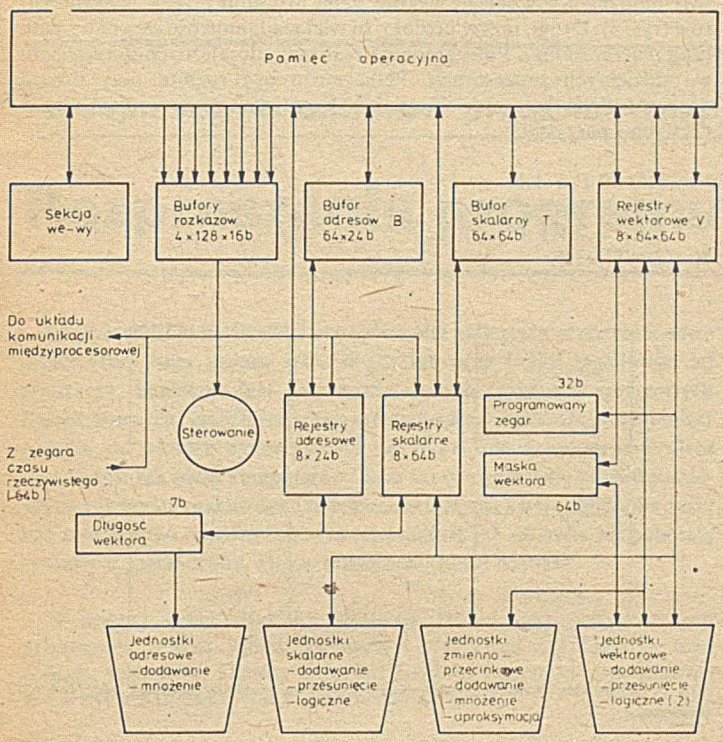


Rys. 1. Cray X-MP – schemat ogólny (– dane, --- sterowanie)

Zasada działania poszczególnych elementów procesora została już dokładnie opisana w wymienionym artykule *Informatyki*.

Poniżej przedstawiono tylko najistotniejsze zmiany w porównaniu z procesorem Cray-1:
– czas cyklu zegara skrócono z 12,5 ns do 9,5 ns,
– liczbę portów do pamięci zwiększono z 1 do 4 (2 dla odczytu i 1 dla zapisu wektorów oraz 1 dla niezależnych operacji we-wy,
– liczbę jednostek funkcjonalnych powiększono o 2 (z 12 do 14),
– rozmiar bufora paczek rozkazów podwojono (z 256 do 512 paczek),

– do listy 128 rozkazów dodano kilka nowych, umożliwiających dostęp do większej niż w Cray-1 pamięci oraz odczyt i zapis wektorów, których elementy są rozmieszczone w pamięci w sposób nieregularny.



Rys. 2. Struktura procesora Cray X-MP

Szybkość operacji zmiennoprzecinkowych jest następująca:

- dodawanie – 6 cykli (57 ns),
- mnożenie – 7 cykli (66,5 ns),
- aproksymacja odwrotności – 14 cykli (133 ns).

Arytmetyka całkowitoliczbowa jest wykonywana na argumentach 24- lub 64-bitowych. Format zmiennoprzecinkowy zawiera 48-bitową mantysę i 15-bitową cechę. Znaki zapisywane są w kodzie ASCII.

Jak podano w tabeli 1, w zależności od modelu, rozmiar pamięci operacyjnej wynosi od 1 do 16 M słów 64-bitowych, przy czym wszystkie modele mają możliwości rozbudowy. Duży rozmiar pamięci pozwala na zrezygnowanie z czasochłonnych i nieraz bardzo wymyślnych technik programowania, których celem jest rozwiązanie problemu obliczeniowego przy małej pamięci. Zastosowane dodatkowe układy logiczne pozwalają na korekcję pojedynczych i detekcję podwójnych przekłamań w słowie pamięci (SECDED, ang. *Single Error Correction, Double Error Detection*). Czas dostępu do pamięci wynosi 34 ns w modelach X-MP/4 (pamięć bipolarna w technologii ECL) i 68 ns w pozostałych (technologia MOS).

Pamięć jest podzielona na moduły (ang. *bank*) o rozmiarze zależnym od modelu (tab. 1). Przeplot (ang. *interleaving*) umożliwia niezależny i równoczesny dostęp do wszystkich modułów w tym samym cyklu zegara. Przypadające na każdy procesor cztery niezależne porty do pamięci oraz krótszy czas dostępu sprawiają, że szybkość przesłań procesor–pamięć (ang. *memory bandwidth*) wzrosła w porównaniu z maszyną Cray-1 kilkakrotnie. Wspólny dostęp do pamięci wymaga rozwiązania problemu konfliktów, wynikających z prób równoczesnego dostępu do tego samego modułu. W maszynie Cray X-MP przewidziano do tego celu odpowiedni układ, wbudowany do jednostki pamięci. Szereguje on zgłoszenia i minimalizuje powstające opóźnienia.

Układ komunikacji międzyprocesorowej (nie występuje w modelach jednoprocesorowych, tzn. X-MP/1) obejmuje pięć (Cray X-MP/4) lub trzy (Cray X-MP/2) klastry (ang. *cluster*) współdzielonych (dostępnych przez wszystkie procesory) rejestrów, umożliwiających łączność między procesorami i synchronizację ich pracy. Każdy klaster zawiera osiem 24-bitowych rejestrów adresowych, osiem 64-bitowych rejestrów skalarnych i 32 przerzutniki synchronizacji (semafory). Przydziałem klastrów do procesorów zarządza system operacyjny. Klaster może byc przydzielony na raz do wszystkich procesorów w systemie lub tylko do ich części. Każdy procesor ma dostęp do klastra w trybie użytkownika (ang. *user mode*) lub systemowym (ang. *system – monitor mode*). Wszystkie procesory korzystają ze wspólnego, 64-bitowego zegara czasu rzeczywistego.

Nowością, która pojawiła się w systemie Cray X-MP, jest zewnętrzna pamięć półprzewodnikowa SSD. Stanowi ona integralną część komputera i jest przystosowana do dużej szybkości przesyłania danych. SSD jest używana do przechowywania dużych plików, z których programy korzystają wielokrotnie (również z plików roboczych). Dla wygody programisty plik jest przepisywany do SSD pojedynczym poleceniem systemowym, bez potrzeby modyfikowania programu, który z niego korzysta. Pamięć SSD jest również wykorzystywana przez system operacyjny do wymiany zadań (ang. *swapping*) i tymczasowego przechowywania programów systemowych. Rozmiar SSD jest zależny od konkretnego modelu jednostki SSD (tabela 2), przy czym każdy z modeli SSD może współpracować z każdym z modeli jednostki centralnej. SSD jest połączona z jednostką centralną jednym (w modelach X-MP/1 i X-MP/2) lub dwoma (X-MP/4) kanałami o maksymalnej szybkości 1000 MB/s. Czas dostępu do SSD nie przekracza 25 μs. O szybkości SSD i jej kanału może świadczyć wynik przykładowego programu wzorcowego (ang. *benchmark*) analizy strukturalnej. W systemie korzystającym tylko z dysku łączny czas operacji we-wy wyniósł 379 sekund; w systemie korzystającym z SSD czas ten wyniósł tylko 5 sekund (w obu wypadkach czas pracy procesora wynosił 165 sekund). Pamięć SSD może być też dołączona do zewnętrznego układu we-wy. Umożliwia to bezpośrednie przesyłanie danych między oboma układami, bez angażowania pamięci operacyjnej.

Tabela 2. Rozmiar pamięci SSD

Model	Rozmiar w megasłowach	Rozmiar w MB
SSD-31	32	256
SSD-4	64	512
SSD-5	128	1024
SSD-6	256	2048
SSD-7	512	4096

Zewnętrznymi układami we-wy, w zależności od modelu jednostki centralnej, są 2, 3 lub 4 procesory we-wy. Każdy z nich ma (niewielką) pamięć lokalną i dostęp do wspólnego bufora pamięci półprzewodnikowej w technologii MOS, o pojemności 2, 4 lub 8 M słów. Układ we-wy realizuje równoległe operacje dostępu do urządzeń zewnętrznych, którymi są:

- jednostki dyskowe DD-49 i DD-39 (zastąpiły starsze DD-19 i DD-29), o pojemności 1200 MB każda, zapewniające szybkości przesyłania danych odpowiednio 9,6 i 5,9 MB/s, przy średnim czasie poszukiwania sektora odpowiednio 16 i 18 ms;
- jednostki pamięci taśmowej IBM (seria 3420 i 3480);
- wysunięte (ang. *front-end*) komputery komunikacyjne z układami we-wy mogącymi zawierać 7 sprzęgów, niwelujących różnice między systemem Cray a sprzętem innych producentów co do szerokości kanałów, długości słowa, poziomów logicznych i protokołów; komputerami komunikacyjnymi mogą być systemy firm IBM, CDC, DEC, Honeywell, Data General i Unisys;
- stacje robocze Sun lub Apollo dołączone za pośrednictwem adapterów kanałów NSC zamiast wymienionych sprzęgów;
- szybkie urządzenia graficzne, dołączone specjalnym kanałem HSX-1 (szybkość do 100 MB/s).

Powyższe urządzenia (z wyjątkiem ostatniego) są połączone z układem we-wy kanałami trzech typów, o szybkościach odpowiednio 6, 100 i 1000 MB/s. Istnieje możliwość elektrycznego odseparowania komputera komunikacyjnego, przez zastosowanie łącza światłowodowego o długości do 1000 m.

W uzupełnieniu informacji na temat urządzeń zewnętrznych, należy wspomnieć o zastosowanej w modelu X-MP technice przeplotu pamięci dyskowej (ang. *disk striping*) – dyskowym odpowiedniku przeplotu pamięci operacyjnej. Polega ona na rozmieszczeniu pojedynczego pliku użytkownika na kilku (od 2 do 5) dyskach. Kolejne bloki pliku są umieszczane cyklicznie na dyskach, dzięki czemu jest możliwy równoczesny dostęp do kilku bloków.

TECHNOLOGIA I WYGLĄD ZEWNĘTRZNY

W procesorach Cray X-MP zastosowano 16-bramkowe układy scalone, wykonane w technologii bipolarnej ECL (ang. *Emitter Coupled Logic*), szybsze (czasy propagacji rzędu 300-400 ps) i bardziej upakowane niż w modelu Cray-1. Pamięć operacyjna jest wykonana w technologii ECL (Cray X-MP/4) lub MOS (Cray X-MP/1 i 2). Dzięki większej integracji, udało się początkowo (pierwsze modele z 1982 r.) zmieścić maszynę w tej samej cylindrycznej szafie, która stanowiła obudowę modelu Cray-1. Fizycznie, jednostki centralne wyglądają następująco:

- Cray X-MP/1 i Cray X-MP/2 to osiem połączonych kolumn, tworzących łuk 180° i zajmujących w podstawie powierzchnię ok. 2,5 m²,
- Cray X-MP/4 składa się z 12 kolumn, ustawionych w łuk 270°, zajmujących w podstawie ok. 4 m².

Pamięć SSD mieści się w czterech kolumnach, tworzących łuk 90°. Połączona jest z jednostką centralną dwoma powietrznymi „mostkami", w połowie wysokości kolumn. Zewnętrzny układ we-wy wygląda identycznie, lecz jest wolnostojący (połączony z jednostką centralną kablami pod podłogą), w odległości do 5,8 m.

W 1986 r. wprowadzono tańszy, „okrojony" model X-MP/1, oznaczony symbolem X-MP/14 se. Nie ma on pamięci SSD, przez co udało się zmieścić go w całości, łącznie z układem we-wy, w sześciu kolumnach, tworzących łuk 135°. Każda szafa jest otoczona znaną już z modelu Cray-1 „ławeczką", kryjącą układy chłodzenia i zasilacze. Większość zewnętrznych powierzchni pokryto skajem, nadając maszynie estetyczny wygląd.

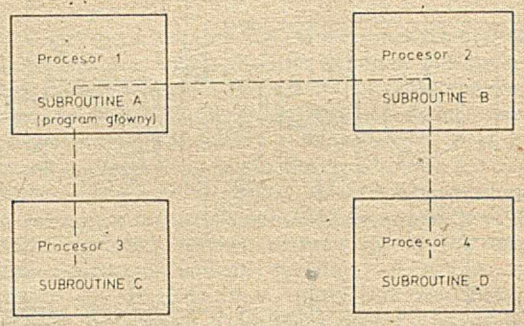
OPROGRAMOWANIE

Cray X-MP może pracować pod kontrolą systemów operacyjnych: COS lub UNICOS. COS, opracowany znacznie wcześniej (dla Cray-1), stanowi zbiór programów rezydujących w pamięci lub na dysku i zarządzających zasobami oraz nadzorujących przetwarzanie prac i operacje we-wy. W środowisku Cray, na typową pracę (ang. *job*) składają się zazwyczaj: kompilacja, ładowanie i wykonanie programu oraz przetwarzanie wyników. COS jest systemem wieloprogramowym. Prace są wykonywane w trybie wsadowym, w dość tradycyjny sposób: sekwencyjne wczytywanie, interpretacja i wykonanie dyrektyw sterujących – podobnie jak np. w systemie OS dla maszyn IBM 360/370. Samo wprowadzenie prac do systemu odbywa się albo za pośrednictwem komputera komunikacyjnego, albo ze stacji wprowadzania prac (lokalnej lub zdalnej). W pierwszym wypadku możliwa jest praca interakcyjna. UNICOS, jako podstawowy system operacyjny modelu Cray-2, omówiony zostanie w drugiej części artykułu.

Podstawowym językiem wysokiego poziomu, używanym przy programowaniu superkomputerów, jest (i pozostanie jeszcze zapewne przez długie lata) Fortran. W systemie X-MP dostępne są dwa kompilatory tego języka: CFT i CFT 77, dostosowane do wersji ANSI 78. Umożliwiają one automatyczną wektoryzację programu i generują optymalny kod wynikowy, zarówno pod względem operacji wektorowych, jak i skalarnych (wektoryzacja polega na przekształceniu programu na taką postać, aby pętle mogły być wykonywane w sposób potokowy, co wymaga pewnej wprawy programisty, w związku z czym cenna jest odpowiednia umiejętność kompilatora). Dostępne są również kompilatory języków Pascal i C, które również umożliwiają operacje wektorowe i posiadają opcję automatycznej wektoryzacji. Nie są one jednak wykorzystywane przez użytkowników w takim stopniu jak Fortran. Przyczyną jest znacznie bogatsze środowisko programowe Fortranu. Asembler CAL jest używany głównie przez programistów systemowych, w języku asemblerowym napisano system COS i inne wcześniejsze programy (jeszcze dla Cray-1), natomiast nowsze oprogramowanie (w tym wersję CAL-a, generującą kod przenośny na wszystkie maszyny Cray) pisze się obecnie głównie w Pascalu.

Cray X-MP jest uniwersalnym systemem wieloprocesorowym, zaprojektowanym szczególnie do pracy wielozadaniowej. Może realizować niezależne lub wzajemnie ze sobą powiązane zadania na kilku procesorach. Istnieje kilka poziomów równoległości obliczeń. Najwyższym jest wieloprocesowość (ang. *multiprocessing*). Polega ona na współbieżnej realizacji oddzielnych procesów przez oddzielne procesory. W najprostszym wypadku, procesami są programy wykonywane niezależnie – każdy w jednym procesorze. Niższym poziomem, już w ramach jednego procesora, jest podział procesu na niezależne zadania (np. gdy na proces składa się wykonanie kilku programów). Kolejne dwa poziomy stanowią cechę charakterystyczną sposobu obliczeń przyjętego w systemie Cray X-MP. Pierwszy, to wielozadaniowość wewnątrz programu (ang. *macrotasking*). Przez zadanie rozumie się w tym momencie część programu, w postaci wydzielonej procedury. System X-MP umożliwia równoległe wykonywanie procedur przez kilka procesorów (rys. 3). Drugi, niższy poziom, to wielozadaniowość wewnątrz pętli (ang. *microtasking*). Polega ona na realizacji niezależnych iteracji pętli w oddzielnych procesorach. Pętla może być rozbita przy dwóch procesorach na dwie oddzielne pętle wykonujące obliczenia dla N nieparzystych i parzystych.

```
      DO 10 I = 1,N
      A(I) = B(I) + C(I)
      D(I) = A(I) + E(I)
10 CONTINUE
```



Rys. 3. Wielozadaniowość w systemie Cray X-MP na poziomie programu przy czterech procesorach

Współbieżnością wykonania zadań steruje opisany wcześniej układ komunikacji międzyprocesorowej. Procesy korzystają ze wspólnego układu semaforów binarnych i rejestrów danych. W celu uniknięcia zakleszczeń zastosowano układ sprzętowej detekcji zakleszczenia w ramach klastra.

PC-ARK
SPÓŁKA Z O.O.
ul. Jaracza 3
00-378 Warszawa
oferuje
CENTRALE TELEFONICZNE „EACT"
– abonenckie, mikroprocesorowe, posiadające homologację i pozwolenia na podłączenie do sieci telekomunikacyjnej Państwa
– dla 24 abonentów wewnętrznych
EACT 24/2 – 2 zespoły miejskie
EACT 24/3 – 3 zespoły miejskie
EACT 24/6 – 6 zespołów miejskich
– dla 48 abonentów wewnętrznych
EACT 24/6 tandem – 10 zespołów miejskich
● INFORMACJE ● INSTALACJE ●
● FACHOWY SERWIS ●
Polecamy również * modemy * telefaxy
Sieć punktów serwisowych na terenie całego kraju zapewnia instalację, usługi gwarancyjne i pogwarancyjne.
tel. 26 09 09, 26 27 94, 26 41 18
teleks 816962 pc pl
EO/1271/88

WACŁAW ISZKOWSKI
Instytut Informatyki
Politechnika Warszawska

POLSKIE TOWARZYSTWO INFORMATYCZNE

Przyspieszanie obliczeń przez ich rozpraszanie (2)

Rozpraszanie programu

Po wyczerpaniu możliwości przyspieszenia obliczenia przez rozbudowę architektury oraz zmianę technologii (lub przy braku możliwości jej realizacji), konieczne jest poszukanie nowych rozwiązań sprzętowych i programowych, umożliwiających zrównoleglenie obliczeń. Natychmiast nasuwa się pomysł wykorzystania na rzecz jednego programu wielu procesorów lub wielu komputerów. Tym samym następuje zwielokrotnienie par: strumień rozkazów ze strumieniem danych, a więc obliczenie może być realizowane współbieżnie. Na tej zasadzie już od wielu lat powstają systemy *n*-procesorowe, zwane maszynami klasy MIMD.

W drugiej części artykułu zajmę się omówieniem podstawowych problemów związanych z realizacją i efektywną eksploatacją tego rodzaju systemów. Ogólny przegląd struktur systemów *n*-procesorowych był już prezentowany w *Informatyce* [6].

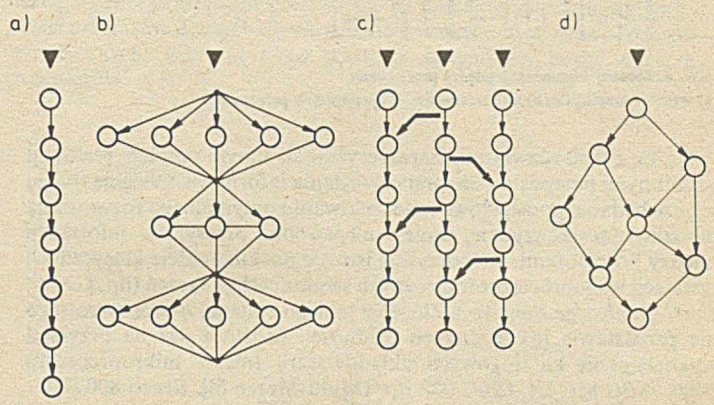
KRYTERIA ROZPRASZANIA

Możliwość zrównoleglenia obliczeń z wykorzystaniem systemów *n*-procesorowych wynika z samej specyfiki algorytmizowanego problemu oraz przyjętych metod podziału obliczenia na pojedyncze – wykonywane już sekwencyjnie – procesy (wątki). Podstawą tego podziału jest przyjęcie wielkości rozdrobnienia[1]) obliczeń (ang. *granularity*). Miarą rozdrobnienia jest współczynnik R/C, w którym R jest liczbą jednostek czasu kwantu obliczenia zadania korzystającego z pojedynczego procesora, C – liczbą jednostek czasu zużytych na komunikację tegoż procesu z innymi, nie znajdującymi się w tym samym procesorze. Rozdrobnieniem gruboziarnistym (duża wartość współczynnika R/C) jest podział treści programu na zadania, z których każde jest złożoną czasową procedurą z małym narzutem na komunikację z innymi procedurami. Rozdrobnieniem drobnoziarnistym (mała wartość współczynnika R/C) jest podział zadania na mniejsze części (nawet na poziomie pojedynczych rozkazów) z małym narzutem komunikacyjnym; przy większych założonych procedurach jest wymagana częsta komunikacja z innymi zadaniami (tabela 1).

Tabela 1. Programy o rozdrobnieniu drobno- i gruboziarnistym

Fragment programu zapisany w Algolu 68 z rozdrobnieniem drobnoziarnistym.
par begin $p5:=5$, $p3:=a(x)$, $p1:=x\uparrow 2$ **end**;
$p2 := p1 + p2 + p3$;
W zdaniu rozpoczynającym się od **par** wszystkie instrukcje mogą być wykonane współbieżnie, a więc dla każdej z nich zostanie utworzony osobny proces (wątek). Obliczenie funkcji $a(x)$ może być długotrwałe, ale przypisanie wartości stałej 5 jest jednym rozkazem, a więc stopień rozdrobnienia dla procesu $p5:=5$ jest bliski 1.
Przykładem rozdrobnienia gruboziarnistego może być zadanie (proces) zapisane w języku Ada [8].

Stopniem równoległości algorytmu programu jest potencjalna możliwość podziału na obliczenia wykonywane współbieżnie. Stopień ten, wyrażany liczbą obliczeń możliwych do równoczesnego wykonania, jest przeważnie zmienny w ramach całego programu oraz, co ważniejsze, zależy od stopnia rozdrobnienia przyjętego dla danej metody wykonania obliczenia. Zwiększenie stopnia równoległości programu pociąga za sobą z reguły zmniejszenie stopnia rozdrobnienia. Graficznym obrazem stopnia równoległości programu jest graf przepływu sterowania (grafy te w zależności od potrzeb i autorów są różnie nazywane, np. *CFG – Control Flow Graph, PTE – Program Task Graph, TFG – Task Flow Graph, DFG – Data Flow Graph, DAG – Directed Acyclic Graph* itp.). Przykłady charakterystycznych grafów przepływu sterowania dla algorytmów o różnych stopniach równoległości pokazano na rys. 1.



Rys. 1. Przykłady grafów przepływu sterowania dla różnych stopni równoległości

Przy tej okazji warto wskazać możliwość znacznego zrównoleglenia obliczeń, nawet z natury sekwencyjnych. Wystarczy bowiem przyjąć, że interesuje nas wykonanie tego samego obliczenia dla wielu zestawów danych (a przecież tak właśnie jest w rzeczywistości), co pociąga za sobą powstanie przetwarzania potokowego lub równoległego w skali całego programu.

SYSTEMY N-PROCESOROWE

Wieloprocesorem luźno powiązanym jest sieć kompletnych komputerów połączonych ze sobą podobnie jak sieci lokalne. Istotna różnica polega jedynie na filozofii funkcjonowania takiego układu, zawartej przede wszystkim w zintegrowanym systemie operacyjnym globalnie nadzorującym współbieżne wykonywanie obliczeń. W sieci lokalnej każda ze stacji ma lokalnie funkcjonujący system operacyjny całkowicie niezależny od działania pozostałych; jego dodatkowym zadaniem jest tylko realizowanie poleceń otrzymywanych z łącza szeregowego – łącznie z pakietem danych – oraz przekazywanie poleceń.

Tabela 2. Przykłady wieloprocesorów luźno powiązanych

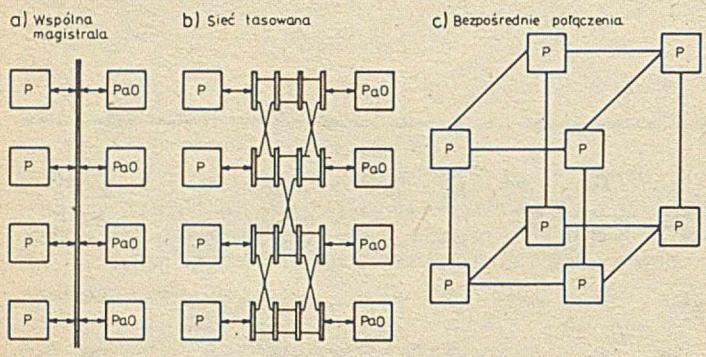
Zintegrowany rozproszony system operacyjny [2] DRAGON SLAYER (Szewc Dratewka) został opracowany dla 16 mikrokomputerów typu IBM PC/XT i AT połączonych ze sobą w sieć lokalną, z protokołem dostępu CSMA (ang. *Carrier Sense Multiple Access*). System ten, zrealizowany na zamówienie armii amerykańskiej jest przeznaczony do wykonywania równoległych programów zapisanych w języku CSP. Zadaniem tych programów jest lokalizacja obiektów przez jakiś lądownik. Najciekawszym jednak elementem tego projektu jest fakt, że w systemie wykorzystywano algorytm szeregowania rozproszonych zasobów, oparty na teorii systemów interakcyjnych, opracowany przez J. Winkowskiego [10].
Na podobnej zasadzie dla kilku mikrokomputerów PC/XT i AT połączonych w sieć Ethernet opracowano rozproszony system przetwarzania wykorzystujący język programowania Martlet – oparty na języku Ada [9].

Wieloprocesorem ściśle powiązanym jest zestaw kilku, kilkunastu lub więcej typowych procesorów, komunikujących się przez wspólnie dostępną pamięć operacyjną. W systemach tych wspólna pamięć może

[1]) Autor używa wyrazu granulacja (przyp. red.).

być wyróżnionym osobnym blokiem lub też może być rozproszona przy poszczególnych procesorach sprzężonych z blokiem lokalnej pamięci operacyjnej oraz pamięci podręcznej. Fizyczna komunikacja między procesorami (oraz dostęp do pamięci wspólnej) może być realizowana przez (rys. 2):

- wspólną magistralę,
- wielopoziomową sieć przełączającą,
- bezpośrednie połączenia w sieci przestrzennej,
- krzyżownicę połączeń.



Rys. 2. Sposoby komunikacji między procesorami
a) wspólna magistrala, b) sieć trasowana, c) bezpośrednie połączenia

Każde z tych rozwiązań charakteryzuje się innym kosztem realizacji oraz innymi przepustowościami przesyłania informacji. Obecnie trwają jeszcze badania podstawowe w poszukiwaniu optymalnego rozwiązania umożliwiającego szybkie, tanie i niezawodne przesyłanie informacji między procesorami. Jednocześnie istnieje już kilkanaście klasycznych instalacji wieloprocesorów o różnych strukturach połączeń (np. *C.nmp, Cm*, Cambridge Ring* itp.). Złożenie takiego systemu *n*-procesorowego nie przedstawia już większych trudności. Możliwe jest na przykład wykorzystanie katalogowych układów serii Intel – mikroprocesora 8086, *iSBC* 86/12*A*, *iSBC* 032 itp. (Multi-Maren [4], Elwro 800 [1]).

Tabela 3. Wieloprocesor BBN Butterfly

Obecnie najżywiej analizowanym jest wieloprocesor BBN Butterfly. Może on mieć od 1 do 256 węzłów. Każdy węzeł (ang. *processor node*) jest osobną kartą zawierającą mikroprocesor MC68000 (lub MC68020 z koprocesorem MC68881), 1 MB pamięci operacyjnej (plus 3 MB na płycie dodatkowej), koprocesor sterownika węzła (PNC – *processor node controller*) zbudowany z układów segmentowych AM2901, szynę we-wy oraz łącze do sieci przełączającej. Siecią tą może być Butterfly Swith (umożliwiająca równoczesne skonfigurowanie dwóch ścieżek połączeń między węzłami, z szybkością przesyłania 32 Mb/s) lub sieć typu Omega. Wykorzystując przełącznik, każdy procesor oprócz dostępu do własnej lokalnej pamięci (w czasie 2 μs) ma dostęp do pamięci każdego innego węzła (w czasie około 6 μs). Możliwe jest również inne skonfigurowanie systemu przez dołączenie osobnych bloków pamięci tworzących wspólną pamięć dla procesorów. Procesor BBN jest przez sieć Ethernet dołączany do systemów SUN lub VAX. Oprogramowaniem wieloprocesora jest system Chrysalis podobny do Unixa oraz język C.

OPROGRAMOWANIE WIELOPROCESORA

Podstawowym problemem w oprogramowaniu wieloprocesora jest zrealizowanie wzajemnego wykluczania w dostępie do wspólnej magistrali oraz pamięci. W tym celu w procesorach muszą istnieć specjalne rozkazy (tab. 4), niezbędne do realizacji bardziej złożonych operacji synchronizacji i komunikacji między procesami zlokalizowanymi w różnych procesorach, a także do implementacji rozproszonego typu kolejkowego przeznaczonego do szeregowania procesów działających w wieloprocesorze. Sposób realizowania tych operacji ma istotny wpływ na efektywność działania wieloprocesora (co zostanie pokazane dalej). Następnie operacje te są częścią zintegrowanego rozproszonego systemu operacyjnego zarządzającego wszystkimi zasobami wieloprocesora.

Użytkownicy wieloprocesorów muszą dysponować językiem wysokiego poziomu. Typowym jest jeden ze znanych języków programowania równoległego: Concurrent Pascal, Edison, Modula-2, CSP, Ada, Chill itp. [3, 7]. Zgodnie z zasadami programowania w tych językach, naturalne jest przypisanie każdemu z procesorów jednego lub więcej procesów (zadań – w Adzie), przy wykorzystaniu mechanizmu komunikacji lub monitorów zlokalizowanych w pamięci wspólnej do wymiany informacji. Tak też jest w rzeczywistości, przy czym już nawet pobieżna analiza tych języków ujawnia wiele niedogodności z punktu widzenia systemu wieloprocesorowego. Brakuje na przykład możliwości zapisu w samym języku sposobu przydziału statycznego lub dynamicznego procesu (zadania) do procesora. Trudne jest również zaimplementowanie mechanizmu synchronizacji między procesami (a tym samym procesorami). Opracowuje się więc nowe języki tego rodzaju, ze zmodernizowanymi mechanizmami komunikacji i zapisem rozproszenia procesów. Jednocześnie języki te umożliwiają programowanie obiektowe.

Tabela 4. Rozkazy obsługi systemu *n*-procesorowego

{} – oznacza nieprzerywalne wykonanie treści danego rozkazu.
Blokowanie dostępu do wspólnej szyny:
Lock
UnLock
Równoczesna wymiana wartości:
Exchange (adres, rejestr)
{ rej := PaO [adres];
PaO [adres] := rejestr;
rejestr := rej}
Testowanie z ustawieniem wartości:
Test&Set (adres, nr_bitu)
{ rej := PaO [adres];
PaO [adres] .nr_bitu := 1;
Kod_warunku := rej.nr_bitu}
Porównywanie z wymianą wartości:
Compare&Swap (adres, rej_st, rej_nowy)
{ rej := PaO [adres];
IF rej = rej_st
THEN
PaO [adres]: = rej_nowy;
Kod_warunku := 1;
ELSE
rej_st := rej;
Kod_warunku := 0
END }
Równoczesne pobranie i dodawanie na wspólnym słowie pamięci:
Fetch&Add (adres, inkrement)
{ rej := PaO [adres];
PaO [adres] := PaO [adres] +inkrement;
RETURN rej
Rozkaz ten jest realizowany częściowo w sieci przełączającej i umożliwia równoczesny dostęp do pamięci wspólnej wielu procesorów w ten sposób, że każdy z nich uzyskuje dotychczasową wartość wskazanego wspólnego słowa zmodyfikowanego danym przez niego przyrostem (ang. increment), a w słowie tym jest umieszczana nowa wartość będąca sumą wartości dotychczasowej i wszystkich przyrostów, np.:
– we wskazanym słowie o adresie xyz znajduje się wartość 20
– w procesorze–1 wydano rozkaz Fetch&Add (xyz, 3)
– w procesorze–3 wydano rozkaz Fetch&Add (xyz, 5)
– w procesorze–6 wydano rozkaz Fetch&Add (xyz, 2)
– procesor–1 otrzyma wartość 23
– procesor–3 otrzyma wartość 25
– procesor–6 otrzyma wartość 22
– we wskazanym słowie pozostanie wartość 30=(20+3+5+2).

Pomimo wielu już lat rozwoju metodyki programowania równoległego oraz strukturalizacji programu przez składnię i semantykę nowo powstających języków, brak jest prostych metod skonstruowania i weryfikacji poprawności programu równoległego. Co więcej, po latach są wykrywane błędy w klasycznych algorytmach implementacji podstawowych operacji lub rozwiązań klasycznych zadań. Istnieją też duże trudności w uruchomieniu i przetestowaniu tego rodzaju niedeterministycznych z założenia programów.

EFEKTYWNOŚĆ DZIAŁANIA

W analizie szybkości obliczeń w systemie wieloprocesorowym bada się przede wszystkim współczynnik przyspieszenia obliczeń $S(n)$ (ang. *speedup*) po zwielokrotnieniu liczby procesorów:

$$S(n) = T(1)/T(n)$$

gdzie $T(n)$ – średni czas obliczenia przy wykorzystaniu n procesorów.

Teoretycznie współczynnik ten powinien być równy liczbie wykorzystywanych w obliczeniu procesorów. W praktyce jednak, wskutek nasycenia w dostępie do wspólnych obszarów pamięci na poziomie wspólnej magistrali systemu oraz na poziomie oprogramowania, współ-

czynnik ten maleje. Zwiększenie jego wartości jest na przykład możliwe przez umieszczanie w pamięci lokalnej danego węzła całego kodu wszystkich procesów, które mogą być wykonywane w danym procesorze. W ten sposób kosztem zwiększenia zajętości pamięci uzyskuje się znaczne przyspieszenie obliczeń.

Na poziomie sprzętu wieloprocesorowego współczynnik przyspieszenia obliczeń jest opisany zależnością

$$S(n) = \frac{R/C * n}{R/C + M(n-1)/2}$$

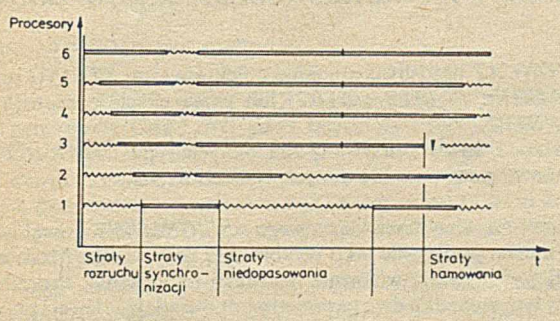
gdzie R/C jest współczynnikiem rozdrobnienia, M – liczbą procesów programu, n – liczbą procesorów.

Analiza tej funkcji (uzyskanej przy założeniu jednakowego czasu wykonywania obliczenia dla każdego z procesów i pojedynczej linii komunikacji między procesorami) prowadzi do następujących wniosków:
– dla małej wartości współczynnika rozdrobnienia R/C w stosunku do wartości $M/2$ współczynnik przyspieszenia będzie mniejszy niż 1,
– wartość współczynnika przyspieszenia zwiększa się proporcjonalnie do wzrostu liczby procesorów tylko przy dużej wartości R/C i małych wartościach M i n (niewielka liczba zadań wykonywana na małej liczbie procesorów),
– przy dalszym zwiększaniu liczby procesorów (duże wartości n) wartość współczynnika przyspieszenia staje się proporcjonalna do $(R/C)/M$, czyli nie zależy od liczby procesorów, przechodzi w stan nasycenia.

Podobne analizy można przeprowadzić przy nieco dokładniejszym opisie modelu obliczenia w systemie wieloprocesowym, uzyskując podobne jakościowo, a odmienne ilościowo wnioski. Z tych badań jednoznacznie wynika, że stosowanie coraz większej liczby procesorów powoduje coraz większe straty w możliwej do uzyskania z nich mocy przetwarzania.

Kolejne straty powstają na poziomie oprogramowania podstawowego z powodu nasycenia programowego związanego z czasowym blokowaniem dostępu do wspólnej pamięci operacyjnej lub określonego portu dostępu w pamięci lokalnej. Przyczyną tych opóźnień są konieczne wzajemne wykluczenia w dostępie do słów pamięci w chwili realizowania synchronizacji, komunikacji lub operacji na kolejkach procesów. Wspomniane już wcześniej problemy realizowania tych operacji wiążą się z przepustowością fizycznego powiązania procesorów między sobą i z modułami pamięci operacyjnej. Również operacje tworzenia, zawieszania, aktywowania oraz likwidowania procesów są dość czasochłonne oraz wymagają dostępu do wspólnych danych i realizacji operacji na rozproszonych kolejkach.

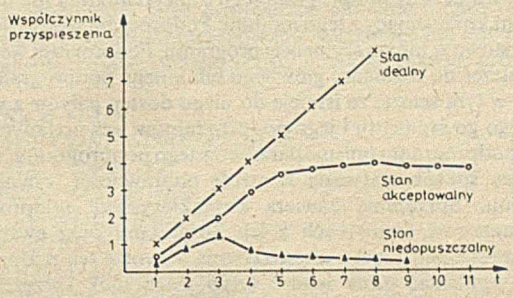
Do oszacowania potencjalnych możliwości przyspieszenia obliczeń w systemie wieloprocesorowym wprowadzono miarę SYPS (ang. *SYnchronizations Per Second*) lub MSYPS (ang. *Milions of SYnchronizations Per Second*). Przykładowo dla procesora z czasem cyklu 100 ns współczynniki te są odpowiednio równe 10^7 SYPS i 10 MSYPS. Dla rozkazów jednocyklowych n-procesor ma moc $10*n$ MIPS, ale zawsze osiągnie tylko 10 MSYPS, co oczywiście ogranicza zwiększanie mocy przetwarzania przez zwiększanie liczby procesorów. Poprawienia tych zależności należy poszukiwać w zwiększaniu rozdrobnienia, zastosowaniu droższych systemów wiążących procesory oraz w równoważeniu systemu przez zredukowanie czasu cyklu procesorów.



Rys. 3. Straty związane z przyspieszaniem obliczeń

Również dokładniejsze analizy organizacji typowych algorytmów użytkowych ujawniają w przebiegu obliczeń rozproszonych straty związane z (rys. 3):
– początkową bezczynnością procesorów, powodowaną koniecznością rozdzielenia dla nich podproblemów do obliczania (straty rozruchu),
– oczekiwaniem na wyniki częściowe przez jeden z procesów, spowodowane nierównomiernym czasowo podziałem obliczenia na zadania między procesy (straty synchronizacji),
– brakiem gotowych do wykonywania procesów, spowodowanym innym stopniem równoległości algorytmu w stosunku do liczby procesorów lub złym podziałem na grupy procesów przydzielanych do tego samego procesora (straty niedopasowania),
– opóźnieniem po znalezieniu wyniku obliczeń przez jeden z procesów, powodowanym koniecznością zatrzymania równolegle biegnących obliczeń (straty hamowania).

Suma tych wszystkich strat jest przyczyną znacznego spadku efektywności systemu wieloprocesorowego, mierzonego współczynnikiem przyspieszenia. Na podstawie wielu analiz można z grubsza podać jego zależność od liczby procesorów i sposobu ich powiązań [6]. Przedstawione na rys. 4 zależności wartości współczynnika przyspieszenia od liczby wykorzystywanych procesorów są charakterystyczne dla większości przeprowadzanych eksperymentów praktycznych oraz rozważań teoretycznych. Inne są tylko wartości optymalnej liczby użytych procesorów.



Rys. 4. Zależność współczynnika przyspieszenia od liczby procesorów

Reasumując, wieloprocesory są dość tanie i łatwo dostępne. Ich efektywne wykorzystanie wymaga jednak jeszcze znacznych badań w doborze najefektywniejszej struktury, specjalnie zaprojektowanego systemu operacyjnego oraz dużych umiejętności w zakresie opracowania algorytmu – odpowiednio rozpisanego na obliczenia dla każdego procesora. Do ich niewątpliwych zalet należy:
- duża elastyczność funkcjonowania z możliwością pełnego wykorzystania właściwości każdego procesora,
- potencjalnie możliwa duża pojemność pamięci operacyjnej rozdzielonej lokalnie między procesory,
- znaczna przepustowość operacji we-wy przypisanych lokalnie do każdego z procesorów, w tym również możliwość wykorzystania ogromnych pamięci masowych,
- wielość zastosowań wynikająca z elastycznej struktury.

LITERATURA

[1] Cellary W.: Systemy wielomikroprocesorowe. Informatyka, nr 11–12, 1986, nr 1, 1987
[2] Friedlander C. B., Wedde H. F.: Distributed Processing under the DRAGON SLAYER Operating System. Proc. 1986 Intern. Conf. on Parallel Processing, Penn State University, 19–22 August 1986
[3] Iszkowski W., Maniecki M.: Programowanie współbieżne. WNT, Warszawa 1982
[4] Iszkowski W.: Performance Evaluation for a Dependency Graph Computations on a Multi Processor System. Research Reports, No. 56, Instytut Informatyki, Politechnika Warszawska, 1983
[5] Iszkowski W.: Implementation of Monitor Operation. MICROSYSTEM'85, Tabor, 1985
[6] Iszkowski W.: Architektura systemów n-mikroprocesorowych, Informatyka, nr 7–8, 1986
[7] Madey J.: Ockham, Edison, Joyce i inni, czyli o programowaniu współbieżnym. IV Jesienna Szkoła PTI, Mrągowo 1987
[8] Pyle I. C.: Ada. WNT, Warszawa, 1986
[9] Shoja G. C., Gurr R. G.: Parallel Processing in Local Networks. Proc. 1986 Intern. Conf. on Parallel Processing, Penn State University, 19–22 August 1986
[10] Winkowski J.: Protocols of Accessing Overlapping Sets of Resources. Information Processing Letters, Vol. 12, No. 5, 1981.

JACEK SKRZYMOWSKI
Instytut Technologii Budowy Maszyn
Politechnika Wrocławska

Nowa norma Fortranu (2)

W drugiej części artykułu omówiono podprogramy, moduły i instrukcje sterujące.

PODPROGRAMY

Rozróżnia się następujące rodzaje podprogramów: zewnętrzne, modułowe i wewnętrzne. Oprócz tego istnieje jeszcze blok sprzężenia podprogramu (ang. *subprogram interface*). Są dwa zasadniczo różne rodzaje podprogramów: funkcje (*FUNCTION*) i procedury (*SUBROUTINE*).

Podprogram zewnętrzny nie jest zawarty tekstowo w żadnym innym segmencie. Może on być przygotowany w Fortranie lub w innym języku. Podprogram modułowy jest definiowany wewnątrz pewnego modułu. Może go wywołać inny podprogram tego samego modułu lub dowolny inny segment korzystający z tego modułu. Podprogram wewnętrzny jest zawarty tekstowo w innym segmencie programu. Podprogram wewnętrzny może należeć do segmentu głównego lub innego podprogramu. Jest on lokalny w tym sensie, że istnieje do niego dostęp jedynie z wnętrza obejmującego go segmentu i jego podprogramów wewnętrznych. Blok sprzężenia podprogramu opisuje parametry tego podprogramu, umożliwiając w ten sposób statyczną kontrolę poprawności wywoływania podprogramu. Sprzężenie zawiera charakterystykę podprogramu: nazwy parametrów formalnych z ich atrybutami oraz ewentualnie operator, przez który można alternatywnie wywołać ten podprogram. W danym segmencie może istnieć tylko jeden blok sprzężenia dla danego podprogramu. Poniżej podano przykłady ilustrujące wymienione pojęcia:

- **podprogram modułowy**

```
MODULE ZNAK
CONTAINS
INTEGER FUNCTION SIGNUM (I)
  IF (I<0) THEN
  SIGNUM = -1
ELSE IF (I = 0) THEN
  SIGNUM = 0
ELSE
  SIGNUM = +1
END IF
END FUNCTION SIGNUM
END MODULE
```

- **procedura wewnętrzna**

```
SUBROUTINE EXTERN
  REAL A, B, C
  ....
  CALL RESET
  ....
  CONTAINS
  SUBROUTINE RESET
    REAL X
    A = 0.0
    B = 0.0
    C = 0.0
    ....
  END SUBROUTINE RESET
END SUBROUTINE EXTERN
```

W powyższym przykładzie procedura zewnętrzna *EXTERN* zawiera procedurę wewnętrzną *RESET*, m.in. zerującą zmienne *A, B, C* procedury *EXTERN*. Podprogramy wewnętrzne znajdują się tekstowo na końcu segmentu, po słowie *CONTAINS*.

- **blok sprzężenia podprogramu**

```
SUBROUTINE INVERSE (A, FN)
  REAL A
  INTERFACE
  FUNCTION FN (B)
  REAL FN, B
  END INTERFACE
  ...
END SUBROUTINE
```

W tym przykładzie blok sprzężenia określa, że drugi argument procedury ma być funkcją typu *REAL* o jednym argumencie typu *REAL*.

Funkcje i procedury mogą być w Fortranie 88 wywoływane rekurencyjnie. W tym przypadku muszą one zawierać w nagłówku słowo *RECURSIVE*. Słowo to musi być podane także wtedy, gdy podprogram wywołuje sam siebie pośrednio, przez inne podprogramy.

```
RECURSIVE INTEGER FUNCTION ALFA (A, B) RESULT (ALF)
  INTEGER A,B
  ...
  ALF = ALFA (A-1, B-1)
  ...
END FUNCTION ALFA
```

Słowo kluczowe *RESULT* określa nazwę zmiennej zawierającej wynik funkcji wywoływanej rekurencyjnie. Każde pojawienie się nazwy funkcji w jej części wykonywalnej jest traktowane jako rekurencyjne wywołanie tej funkcji. Jeśli nie podano by słowa *RESULT*, to zmienna wynikowa byłaby identyczna z nazwą funkcji i każde wystąpienie tej nazwy dotyczyłoby zmiennej wynikowej, nie będąc rekurencyjnym wywołaniem funkcji.

Programista może zdefiniować dla funkcji operator, przez który możliwe będzie alternatywne wywołanie tej funkcji. Możliwe jest to jednak tylko w wypadku, gdy funkcja ma jeden lub dwa argumenty. Operator, który może być ciągiem dowolnych znaków alfabetu definiuje się przez użycie słowa kluczowego *OPERATOR* w nagłówku funkcji. Każde wystąpienie zdefiniowanego operatora z argumentami o tych samych atrybutach co parametry formalne funkcji, traktuje się jako wywołanie tej funkcji.

Zamiast definiowania własnych operatorów można nadać operatorowi standardowemu nowe znaczenie, używając jego symbolu w opcji *OPERATOR* dyrektywy *FUNCTION*. Umożliwia to stosowanie tych samych operatorów do analogicznych działań dla różnych typów danych, np.:

```
PROGRAM MAIN
...
CONTAINS
TYPE (STRING (*)) FUNCTION CONCAT (S1, S2) OPERATOR
(//)
...
```

Funkcja *CONCAT* umożliwia spinanie obiektów typu *STRING* przez wywołanie *CONCAT* (STR1, STR2) lub przez użycie operatora //, np.: *STR1//STR2*.
Operator standardowy // używany dla danych typu *CHARACTER* zdefiniowano tu ponownie dla danych typu *STRING*.

Poprzez umieszczenie słowa kluczowego *ASSIGNMENT* w nagłówku procedury można ją określić jako wykonującą przypisanie. Może ona być później wywoływana w formie instrukcji przypisania. Procedura musi mieć w tym wypadku dwa parametry odpowiadające lewej i prawej stronie instrukcji przypisania, np.:

```
SUBROUTINE ASSIGN CHAR TO STRING (STR, CHAR) ASSIGNMENT
    TYPE (STRING (*)) :: STR
    CHARACTER (LEN=*) :: CHAR
    STR % LENGTH = LEN (CHAR)
    STR % TEXT = CHAR
END SUBROUTINE
```

Powyższa definicja umożliwia przypisanie obiektowi typu strukturalnego *STRING* wartości typu *CHARACTER*, np.:

```
STR1 = 'JanB'
```

Kilka różnych podprogramów można wywoływać przy użyciu tej samej nazwy. Rozróżnienie, który podprogram jest w rzeczywistości wywoływany, następuje na podstawie analizy listy argumentów. Dla funkcji wewnętrznych zasada ta jest znana już z poprzedniej normy, np. jedna standardowa funkcja wewnętrzna *MIN* działa dla różnych typów argumentów.

Przy deklarowaniu parametrów formalnych podprogramów można, używając atrybutu *INTENT*, podać ich zamierzone użycie. *INTENT (IN)* określa dany parametr jako wejściowy, *INTENT (OUT)* jako wynikowy, *INTENT (INOUT)* określa, że parametr może być jednocześnie wejściowy i wynikowy, np.:

```
SUBROUTINE TRANSFER (FROM, TO)
REAL, INTENT (IN) :: FROM
REAL, INTENT (OUT) :: TO
```

Dla parametrów formalnych wprowadza się także atrybut *OPTIONAL*. Dla parametrów posiadających ten atrybut nie jest konieczne podawanie argumentów aktualnych podczas wywołania podprogramu.

Podczas wywoływania podprogramu można powtórzyć nazwy parametrów formalnych, np.:

```
CALL ASSIGN CHAR TO STRING (STR = STRING1, CHAR = 'Jan')
```

Zaletą tego zapisu jest to, że parametry aktualne nie muszą być podawane w kolejności występowania parametrów formalnych, a argumenty opcjonalne mogą zostać po prostu pominięte.

MODUŁY

Jako nowy niewykonywalny typ segmentu wprowadzono moduł. Zawiera on definicje, deklaracje i specyfikacje, które z założenia mają być dostępne w innych segmentach. Obejmują one definicje obiektów globalnych, deklaracje typów, definicje podprogramów i bloki sprzężeń podprogramów. Moduły zastępują funkcje dyrektywy *COMMON* i segmentu *BLOCK DATA*, nie odwołując się przy tym bezpośrednio do obrazu pamięci operacyjnej. Po zdefiniowaniu modułu można się do niego odwoływać przez dyrektywę *USE*, np.:

```
MODULE DATA
    INTEGER, ARRAY (100) :: I
    REAL, ARRAY (10, 10) :: R
    CHARACTER (LEN=100) :: TEXT
END MODULE DATA
SUBROUTINE CALLING
    USE DATA, ONLY : R
    ...
    X = R (J,K)
    ...
END SUBROUTINE CALLING
```

W tym przykładzie moduł *DATA* zawiera deklaracje trzech obiektów. W procedurze *CALLING* przez użycie dyrektywy *USE* uzyskuje się dostęp do tego modułu; zadeklarowane w module zmienne mogą być od tej chwili używane, jeśli ich użycie nie zostało ograniczone. W powyższym przykładzie procedura *CALLING* ma dostęp tylko do tablicy *R*, gdyż możliwości dostępu zostały ograniczone do tej zmiennej przez dodanie słowa *ONLY*.

Dalsze możliwości określania praw dostępu do obiektów modułu zapewniają atrybuty *PRIVATE* i *PUBLIC*. Do obiektów z atrybutem *PRIVATE* nie ma dostępu spoza modułu. Obiekty z atrybutem *PUBLIC* mogą być używane w innych segmentach po użyciu dyrektywy *USE*. Poniżej podano przykład modułu zawierającego jedynie obiekty:

```
MODULE DATA PROG
    REAL A (10), B, C (20, 20)
    INTEGER, INITIAL :: I=0
    INTEGER, PARAMETER, PRIVATE :: J=10
    COMPLEX D(J, J)
END MODULE
```

Do tak zdefiniowanego modułu można odwołać się dyrektywą *USE* trzema sposobami.

```
USE DATA PROG
```

W pierwszym wypadku można mieć dostęp do wszystkich obiektów z wyjątkiem *J*, gdyż ten został zadeklarowany jako *PRIVATE*.

```
USE DATA PROG, ONLY: A, B, D
```

W drugim przykładzie dostęp jest możliwy tylko do zmiennych *A*, *B*, i *D*.

```
USE DATA PROG, A = > ALFA, D = > DELTA
```

W ostatnim wypadku zmienia się nazwy importowanych zmiennych *A* i *D* na *ALFA* i *DELTA*. Jest to przydatne w sytuacji, gdy w segmencie istnieją już inne obiekty o nazwach *A* i *D*.

Wiele programów musi przetwarzać globalne tablice o znacznych rozmiarach, których zapotrzebowanie na pamięć nie jest znane przed wykonaniem. Poniżej podano prosty przykład takiego programu:

```
PROGRAM MAIN
    CALL CONFIGURE ARRAYS
    CALL COMPUTE
END PROGRAM MAIN
MODULE WORK ARRAYS
    INTEGER N
    REAL, ALLOCATABLE, SAVE :: A(:), B(:, :), C(:, :, :)
END MODULE WORK ARRAYS
SUBROUTINE CONFIGURE ARRAYS
    USE WORK ARRAYS
    READ (INPUT, *) N
    ALLOCATE (A(N), B(N, N), C(N, N, N))
END SUBROUTINE CONFIGURE ARRAYS
SUBROUTINE COMPUTE
    USE WORK ARRAYS, ONLY: A, B, C
    ...
END SUBROUTINE COMPUTE
```

Procedura *CONFIGURE ARRAYS* przydziela dynamicznie pamięć tablicom *A*, *B* i *C* instrukcją *ALLOCATE*, i od tej chwili można nimi operować. Przetwarzanie tablic odbywa się w procedurze *COMPUTE*. Wszystkie podprogramy, które chcą mieć dostęp do tych tablic, muszą importować moduł *WORK ARRAYS*.

Podczas pisania bibliotek podprogramów można wszystkie sprzężenia do nich zgromadzić w jednym module. Pozwala to na użycie wywołań z nazwami parametrów formalnych i argumentów opcjonalnych. Poza tym wywołania podprogramów mogą być statycznie kontrolowane ze względu na poprawność.

```
MODULE MATBIB
    INTERFACE
    FUNCTION COTAN (X)
    REAL (* *), INTENT (IN) :: X
    REAL (EFFECTIVE PRECISION (X), EFFECTIVE EXPONENT RANGE (X)) COTAN
    END INTERFACE
    ...
END MODULE
```

– Moduł ten pozwala na wywołanie funkcji *COTAN:*

```
USE MATBIB
...
A = COTAN (X=ARCUS)
...
```

INSTRUKCJE STERUJĄCE

Wzorem innych języków programowania wprowadzono instrukcję *CASE*, której zastosowanie ilustruje poniższy przykład:

```
INTEGER FUNCTION SIGNUM (N)
SELECT CASE (N)
    CASE (:-1)
    SIGNUM = -1
    CASE (0)
    SIGNUM = 0
    CASE (1:)
    SIGNUM = 1
    END SELECT
END FUNCTION SIGNUM
```

Przy wykonywaniu instrukcji *SELECT CASE* oblicza się wartość wyrażenia w nawiasach. Wynik jest nazywany indeksem wyboru. Musi on być zgodny ze stałą lub zakresem wyboru jednej z instrukcji *CASE*. W wypadku podania zakresu wyboru *a:b* indeks wyboru musi spełniać warunek:

$a <= \text{indeks wyboru} <= b.$

Dla zakresu wyboru postaci *:a* powinno zachodzić

$a <= \text{indeks wyboru}$

a dla postaci *a:*

$\text{indeks wyboru} <= a.$

Jeżeli żadne z zakresów bądź stałych wyboru instrukcji *CASE* nie są zgodne z indeksem wyboru, to używa się słowa *DEFAULT*, które odpowiada wartości domyślnej.

Instrukcja *DO* może mieć jedną z następujących postaci:

```
etykieta: DO 100, I=1, 10, 1
etykieta: DO 200 (10) TIMES
```

Druga wersja instrukcji *DO* charakteryzuje się brakiem zmiennej sterującej. Etykieta jest dowolnie wybieralną nazwą pętli *DO* i może zostać opuszczona. Jeśli instrukcja *DO* posiada nazwę, to blok *DO* musi zostać zakończony instrukcją

END DO nazwa pętli.

Etykieta oraz zmienna sterująca mogą zostać również opuszczone. Aby usprawnić programowanie pętli, wprowadzono dwie instrukcje:

```
EXIT nazwa pętli
CYCLE nazwa pętli.
```

Nazwę pętli można opuścić. Wykonanie instrukcji *EXIT* prowadzi do opuszczenia bloku *DO*. Wykonanie instrukcji *CYCLE* powoduje zakończenie wykonywania aktualnego obiegu pętli i kontynuuje jej wykonywanie od początku. Ma ona taki sam efekt co skok do ostatniej instrukcji bloku *DO* i służy przede wszystkim polepszeniu stylu programowania.

```
SUM = 0
READ (*, *) N
DO (N) TIMES
    READ (*, *) P, Q
    CALL CALCULATE (P, Q, R)
    SUM = SUM + R
    IF (SUM > SMAX) EXIT
END DO
```

W tym przykładzie pętla zostanie wykonana tyle razy, ile wynosi wartość *N*, z wyjątkiem wypadku, gdy wartość *SUM* przekroczy *SMAX*, co spowoduje przerwanie pętli. Jeśli *N* przybierze wartość 0, to pętla **DO** nie zostanie ani razu wykonana.

PRZESTARZAŁE ELEMENTY JĘZYKA

Przestarzałe elementy języka dzielą się na dwie kategorie, określane jako odrzucane i zanikające. Konstrukcje odrzucane były zbędne w zasadzie już w Fortranie 77, są jednak jeszcze często stosowane. Konstrukcje zanikające będą nadmiarowe w Fortranie 88 ze względu na wprowadzenie nowych konstrukcji językowych.

Nowe programy w Fortranie powinny wykorzystywać tylko te możliwości języka, które nie są odrzucane lub zanikające, zakłada się bowiem, że w przyszłych normach Fortranu mogą one już nie występować. Dla ułatwienia procesu przechodzenia na nową normę kompilatory Fortranu 88 powinny ostrzegać przed użyciem przestarzałych konstrukcji. Do konstrukcji odrzucanych zalicza się konstrukcję *IF* arytmetyczną, zmienne typu *REAL* lub *DOUBLE PRECISION* jako zmienne sterujące pętli *DO*, zagnieżdżone pętle *DO* kończone jedną instrukcją, skok do etykiety po zakończeniu procedury (*RETURN i*), instrukcję *ASSIGN* i wyznaczaną instrukcję *GOTO*.

Pozostałe przestarzałe elementy języka (zanikające) zostaną zachowane nieco dłużej i będą na pewno zawarte jeszcze w następnej normie. Następujące instrukcje odnoszą się do fizycznej organizacji pamięci i dlatego zostały ocenione jako zanikające: *COMMON, EQUIVALENCE, BLOCK DATA*, tablice formalne z przejętą długością (z gwiazdką zamiast ostatniego wymiaru), element tablicy lub stała tekstowa jako parametr aktualny dla tablicy formalnej. Inne konstrukcje zastąpione przez lepsze zostały także ocenione jako zanikające: sztywny format programu źródłowego (ograniczone użycie pozycji 1..6), dyrektywa *ENTRY*, instrukcja skoku obliczanego, dyrektywy *DATA, DIMENSION, DOUBLE PRECISION*, określanie długości zmiennej znakowej przez **n* (zastąpione przez *LEN = n*), różne nazwy dla funkcji standardowych wykonujących te same operacje na różnych typach.

Imprezy

Informatyka w szkole

V konferencja o tej tematyce odbędzie się **w Wałbrzychu w dniach 12–15 września 1989 r.**
Jej organizatorami są: Instytut Informatyki Uniwersytetu Wrocławskiego, RW NOT w Wałbrzychu, Oddział Doskonalenia Nauczycieli we Wrocławiu, Kuratorium Oświaty i Wychowania w Wałbrzychu, ZETO Wałbrzych z siedzibą w Świdnicy. Honorowy patronat objął minister edukacji narodowej, prof. dr hab. Jacek Fisiak.

Konferencja jest adresowana do osób zainteresowanych wprowadzaniem metod informatycznych do procesu nauczania w szkołach różnego typu. Organizatorzy zapraszają przede wszystkim nauczycieli szkół podstawowych, średnich i wyższych, metodyków, pracowników naukowo-badawczych, a także producentów i dostawców sprzętu i oprogramowania dydaktycznego.

Adres organizatorów:

- **w sprawach merytorycznych**
  Instytut Informatyki, Uniwersytet Wrocławski
  ul. Przesmyckiego 20, 51-151 Wrocław, tel. 25-12-71
- **w sprawach organizacyjnych**
  Biuro RW NOT ul. Szmidta 4a, 58–300 Wałbrzych
  tel. 258-83, 253-69; teleks 0745323

Komputeryzacja dydaktyki

Konferencja naukowo-techniczna o tej tematyce odbędzie się w **dniach 28–29 listopada 1989 roku w Domu Technika w Łodzi, Plac Komuny Paryskiej 5a.**
Jej organizatorami są Naczelna Organizacja Techniczna, Instytut Informatyki Politechniki Łódzkiej oraz Polskie Towarzystwo Cybernetyczne. Honorowy patronat objął minister edukacji narodowej, prof. dr hab. Jacek Fisiak.

Celem konferencji jest usystematyzowanie wiedzy na temat komputeryzacji dydaktyki w szkolnictwie wyższym i ponadpodstawowym.

Jej tematyka obejmuje zagadnienia nauczania z wykorzystaniem sprzętu komputerowego z oprogramowaniem dydaktycznym i przedmioty typowo informacyjne, jak bazy danych, systemy operacyjne, języki programowania. Omawiane będą także mikrokomputery jednoukładowe, systemy 8-bitowe, komputery personalne 16- i 32-bitowe, wielodostęp i lokalne sieci komputerowe. Teoretyczna część konferencji zostanie uzupełniona pokazem sprzętu i oprogramowania przeznaczonego do komputeryzacji dydaktyki.

Adres organizatorów:
Wojewódzki Ośrodek Postępu Technicznego RW NOT
Plac Komuny Paryskiej 5a
99–950 Łódź
tel. 33-59-78

JAN BIELECKI
Instytut Informatyki
Politechnika Warszawska

Grafika w Turbo Pascalu 4.0

Jedną z nowości języka Turbo Pascal w wersji 4.0 jest obszerna biblioteka graficzna zawarta w modułach Graph i Crt. Umożliwia ona wygodne programowanie grafiki w sposób niezależny od obranej karty graficznej.

Podprogramy zawarte w module Graph umożliwiają rozpoznanie środowiska graficznego (np. Graph), inicjowanie grafiki pikselowej i znakowej (np. InitGraph, TextMode), przełączanie między trybem graficznym i tekstowym (np. RestoreCrtMode), ustanawianie okienek graficznych i tekstowych (np. SetViewPort i Window), a ponadto: wprowadzanie tekstów (np. Write), wykreślanie tekstów (np. OutText) i wykreślanie podstawowych obiektów graficznych (np. Circle).

Jeśli argumenty podprogramu nie naruszają wymagań składniowych, ale są dobrane niewłaściwie, to wykonanie podprogramu nie wywołuje żadnych skutków. Jeśli argumenty są dobrane właściwie, ale wykonanie podprogramu okaże się niepomyślne, to wykonywanie programu jest kontynuowane. Zbadanie, jaka była przyczyna niepomyślnego wykonania operacji graficznej pozostawia się programującemu. Zadanie to ułatwia funkcja GraphResult, której działanie (przedstawione szczegółowo dalej) może być porównane do działania funkcji IOResult.

Rezultatem funkcji GraphResult jest dana typu integer. Jeśli wykonanie podprogramu było pomyślne, to dana ta ma wartość 0. W przeciwnym razie ma wartość ujemną, identyfikującą przyczynę niepowodzenia. Ponieważ dwukrotne wywołanie funkcji GraphResult (bez wykonania w międzyczasie operacji graficznej) powoduje, że jej drugim rezultatem jest zawsze dana o wartości 0, zaleca się przypisanie pierwszego rezultatu zmiennej pomocniczej (analogicznie do typowego użycia funkcji IOResult).

TRYB TEKSTOWY

Wykonanie każdego programu rozpoczyna się w trybie tekstowym. Znaki wyprowadzane na ekran mogą być białe na czarnym tle, czarne na białym tle, migające i kolorowe. Wyprowadzanie znaków odbywa się za pośrednictwem plików skojarzonych z konsolą.

OKIENKO TEKSTOWE

Po rozpoczęciu wykonania programu domniemanym okienkiem tekstowym jest cały ekran. Lewy górny narożnik ekranu tekstowego ma współrzędne (1,1), a prawy dolny ma współrzędne *(col, lin)*, gdzie *col* jest liczbą kolumn, a *lin* jest liczbą wierszy ekranu; np. dla karty CGA w trybie C40 są to współrzędne (40, 25). Z okienkiem jest związany widoczny kursor, którego położenie może być zmieniane za pomocą podprogramu GotoXY. Zmiana okienka tekstowego może być dokonana za pomocą podprogramu Window. Argumentami tego podprogramu są współrzędne ekranowe. Najmniejsze okienko ma wymiary 1x1 znak. Wszystkie operacje tekstowe dotyczą bieżącego okienka tekstowego, a współrzędne kursora tekstowego są zawsze liczone względem lewego górnego narożnika okienka. Okienko ma wszystkie właściwości ekranu monitora; w szczególności umożliwia przeglądanie pionowe.

Przykład. Przeglądanie pionowe

```
program Scrolling;
uses
  Crt;
begin
  ClrScr;
  Window(1,1,5,3);
  Writeln('Jan');
  Writeln('Izabela')
end.
```

Na ekranie pojawia się wiersz zawierający napis *Izabe* i wiersz zawierający napis *la*. Na skutek przesunięcia okienka tekstowego, napis *Jan* znika.

KOLORY

Wyprowadzanie znaków na ekran może odbywać się w trybie czarno-białym albo kolorowym. Jeśli użyta karta zapewnia wyprowadzanie kolorowe, to mogą być niezależnie określone atrybuty pierwszego planu i tła znaku (np. czerwony znak na zielonym tle). Ponadto można określić atrybut migania znaku. Wszystkie te atrybuty są przypisywane zmiennej TextAttr i dotyczą wszystkich następnie wyprowadzanych znaków. Ustawianie atrybutów odbywa się za pomocą procedur TextColor i TextBackground. Ich argumentami mogą być symbole oznaczające kolory. Na monitorze monochromatycznym sterowanym przez kartę Hercules mają sens jedynie przedstawione w tabeli zestawy argumentów procedur TextColor (pierwszy plan) i TextBackground (tło).

Zestaw argumentów procedur TextColor i TextBackground dla karty Hercules

Pierwszy plan	Tło	Skutek
7	0	Białe znaki na czarnym tle
1	0	Białe podkreślone znaki na czarnym tle
0	7	Czarne znaki na białym tle
0	0	Czarne znaki na czarnym tle (oczywiście niewidoczne)

Przykład. Wyprowadzanie znaków w kolorach

```
program TextColors;
uses
  Crt;
begin
  ClrScr;
  TextColor(Red);
  TextBackground(Green);
  Write('Kajusia')
end.
```

Na monitorze kolorowym sterowanym przez odpowiednią kartę graficzną (np. EGA albo CGA) pojawia się czerwony napis *Kajusia* na zielonym tle.

TRYB GRAFICZNY

Przełączanie systemu do trybu graficznego wymaga wykonania podprogramu InitGraph (jeśli posłużono się modułem Graph) albo wykonania podprogramu GraphMode, GraphColorMode, albo HiRes (jeśli posłużono się modułem Graph3, a w systemie jest zainstalowana karta CGA albo karta stanowiąca jej rozszerzenia, np. EGA). Dalsze rozważania zostaną ograniczone do posłużenia się modułem Graph.

Przykład. Ustanowienie trybu graficznego

```
program Initialize;
uses
  Crt,Graph;
var
  Driver,Mode : integer;
begin
  Driver := Detect;
  InitGraph(Driver,Mode,'');
  PutPixel(0,0,1);
  repeat until KeyPressed;
  CloseGraph
end.
```

Wykonanie programu powoduje automatyczne rozpoznanie środowiska graficznego, przełączenie systemu do trybu graficznego i wykreślenie na ekranie jednego piksela.

OKIENKO GRAFICZNE

Po ustanowieniu trybu graficznego domniemanym okienkiem graficznym jest cały ekran. Lewy górny narożnik ekranu graficznego ma współrzędne (0, 0), a prawy dolny ma współrzędne (GetMaxX, GetMaxY). Zmiana okienka graficznego może być dokonana za pomocą podprogramu SetViewPort. Jego argumentami są współrzędne okienka oraz wyrażenie logiczne, określające czy wykresy wykraczające poza okienko mają być obcinane czy nie. Wszystkie operacje graficzne dotyczą bieżącego okienka graficznego, a współrzędne kursora graficznego są zawsze liczone względem lewego górnego narożnika okienka. Kursor graficzny jest niewidoczny, ale jego bieżące współrzędne mogą być określone za pomocą procedur GetX i GetY. Współrzędne te są wartościami danych typu integer i mogą być zarówno dodatnie, jak i ujemne. Jeśli współrzędna X nie należy do przedziału 0..GetX lub współrzędna Y nie należy do przedziału 0..GetY, to kursor znajduje się poza okienkiem graficznym.

Przykład. Obcinanie na granicach okienka

```
program Clipping;
uses
  Crt,Graph;
var
  Driver,Mode : integer;
begin
  Driver := HercMono;
  Mode := HercMonoHi;
  InitGraph(Driver,Mode,'');
  SetViewPort(1,1,GetMaxX,GetMaxY,ClipOn);
  PutPixel(-1,-1,1);
  repeat until KeyPressed;
  CloseGraph
end.
```

Ponieważ procedurę SetViewPort wywołano z argumentem GlipOn (obcinaj), wykonanie procedury PutPixel nie ma żadnych skutków. Gdyby argument ClipOn zastąpiono argumentem ClipOff (nie obcinaj), w pobliżu lewego górnego narożnika okienka graficznego (ale poza okienkiem) zostałby wykreślony jeden piksel.

WYKREŚLANIE TEKSTÓW

Teksty wyprowadzane na ekran graficzny są na nim wykreślane. teksty mogą być dwóch rodzajów: bitowe i kreskowe. Każdy znak tekstu bitowego jest prostokątem o wymiarze podstawowym 8x8 bitów. Znak taki po powiększeniu nie wygląda zbyt dobrze. Znacznie lepszy efekt wizualny daje powiększony znak kreskowy. Jest on tworzony w sposób wektorowy i czytelność znaku nie zależy od jego rozmiaru. Przewidziano cztery kroje znaków kreskowych: indeksowy, potrójny, bezszeryfowy i gotycki. Znaki tych krojów mogą być nie tylko powiększane, ale mogą być również rozciągane w poziomie i w pionie. Teksty składające się ze znaków dowolnego kroju mogą być wyprowadzane w poziomie (od lewej do prawej) albo w pionie (od dołu do góry). Alfabety krojów obejmują znaki podstawowego kodu ASCII. Kroje kreskowe są przechowywane na dysku i ładowane dynamicznie do pamięci operacyjnej. W każdej chwili w pamięci może znajdować się co najwyżej jeden automatycznie załadowany krój kreskowy. Zapewniono środki umożliwiające ładowanie krojów na żądanie (procedura RegisterBGIFont) oraz instalowanie ich w programie na stałe.

Przykład. Wykreślanie tekstu

```
program Texts;
uses
  Crt,Graph;
var
  Driver,Mode : integer;
begin
  Driver := HercMono;
  Mode := HercMonoHi;
  InitGraph(Driver,Mode,'');
  SetTextStyle(SmallFont,HorizDir,10);
  OutText('Jan ');
  SetTextStyle(GothicFont,HorizDir,8);
  OutText('Ewa ');
  SetTextStyle(TriplexFont,HorizDir,8);
  OutText('Iza');
  Delay(2000);
  CloseGraph
end.
```

Wykonanie programu powoduje wykreślenie trzech tekstów; pierwszy jest wykreślany czcionką o kroju indeksowym, a następnie odpowiednio gotykiem i krojem potrójnym. Znaki tekstu *Jan* są powiększone 10-krotnie, a pozostałe znaki 8-krotnie. Teksty są wyprowadzane w poziomie (HorizDir).

WYKREŚLANIE OBIEKTÓW GRAFICZNYCH

Biblioteka podprogramów graficznych zawiera podprogramy do wykreślania pikseli (np. PutPixel), odcinków (np. Line), łuków (np. Arc), okręgów (np. Circle), elips (np. Ellipse), a ponadto łamanych, wielokątów, słupków itp. Linie mogą być wykreślane jako grube albo cienkie, ciągłe albo przerywane, a obszary mogą być wypełniane wzorami standardowymi i projektowanymi.

Przykład. Wykreślenie wypełnionego okręgu

```
program Objects;
uses
  Crt,Graph;
var
  Driver,Mode : integer;
begin
  Driver := HercMono;
  Mode := HercMonoHi;
  InitGraph(Driver,Mode,'');
  Circle(GetMaxX div 2,GetMaxY div 2,200);
  FloodFill(GetMaxX div 2,GetMaxY div 2,1);
  repeat until KeyPressed;
  CloseGraph
end.
```

Wykonanie programu powoduje wykreślenie okręgu o promieniu 200 pikseli i środku w pobliżu środka ekranu, a następnie wypełnienie go wzorem domniemanym. Wykreślony obiekt jest obcięty na granicach okienka graficznego.

KOLORY

Wykreślane obiekty graficzne mogą być czarno-białe albo kolorowe. Wykreślanie czarno-białe dotyczy zazwyczaj kart graficznych nie uwzględniających koloru, jak np. karta Hercules. Oczywiście w takim wypadku kolor oznaczony jako biały zależy od użytego luminoforu i może być w istocie np. zielony albo bursztynowy.

Jeśli jest możliwe wyświetlanie kolorowe (np. karta EGA albo CGA), to podczas instalowania sterownika graficznego można określić tryb jego pracy, a tym samym dostępną paletę kolorów. Kolory palety (nie istnieje ona p. dla karty Hercules), są ponumerowane od 0 do GetMaxColor. Kolory te mogą być ustalone (np. karta CGA) albo dowolnie zmieniane (np. karta EGA). Jeśli kolory mogą być zmieniane, to przypisanie pozycji palety obranego koloru odbywa się za pomocą podprogramu SetPalette. Wybór numeru koloru do wykreślania obiektów zapewnia podprogram SetColor. Po wykonaniu podprogramu SetColor(n), wszystkie obiekty są wykreślane w kolorze związanym (sprzętowo albo programowo) z pozycją o numerze *n* palety kolorów. Programowe przypisanie tej pozycji innego koloru powoduje natychmiastową zmianę koloru rozpatrywanych obiektów.

Przykład. Wyświetlanie w kolorach

```
program GraphColors;
uses
  Crt,Graph;
var
  Driver,Mode : integer;
  Palette : PaletteType;
  i : shortint;
  Drop : char;
  Color : byte;
begin
  Driver := EGA;
  Mode := EgaLo;
  InitGraph(Driver,Mode,'');
  if GraphResult <> grOK then Halt;
  SetPalette(1,Red);
  SetColor(1);
  Line(0,0,GetMaxX,0);              { Red line }
  repeat until KeyPressed;
  Drop := ReadKey;
  SetPalette(1,Green);              { Green line }
  OutTextXY(0,0,'Press a key');     { Green text }
  repeat until KeyPressed;
  Drop := ReadKey;
```

```
GetPalette(Palette);
Color := GetColor;
RestoreCrtMode;
Writeln('Palette ');
with Palette do
  for i := 0 to Size - 1 do
    Writeln('No. ',i :2,' = ',Colors[i]);
end.
```

Przykładowy program na wydruku ilustruje zasadę posługiwania się paletą kolorów karty EGA. Po ustanowieniu trybu graficznego i związaniu z pozycją nr 1 palety koloru czerwonego (Red) jest wykreślany poziomy odcinek linii prostej. Po naciśnięciu dowolnego klawisza klawiatury, następuje zmiana koloru związanego z pozycją nr 1 palety na zielony (Green). Powoduje to, że zarówno już wykreślony odcinek, jak i tekst *Press a key* jest wyświetlany w tym kolorze. Na zakończenie wykonania programu następuje przełączenie systemu do trybu tekstowego i wyprowadzenie tablicy określającej identyfikatory kolorów przyporządkowane poszczególnym pozycjom palety.

Ze świata

Problemy stosowania Ady

Ciągle wzrastający nacisk ze strony Departamentu Obrony USA powoduje, że Ada zyskuje coraz większą popularność wśród przedsiębiorstw związanych z przemysłem zbrojeniowym. Ku zdziwieniu krytyków języka, zyskuje on przychylność nowych użytkowników. Doświadczenia zdobyte w kolejnych projektach utwierdzają personel kierowniczy wielu firm w przekonaniu, że Ada jest obecnie najlepszym językiem projektowania dużych systemów.

ZASTOSOWANIA WOJSKOWE

W związku z rozszerzającym się kręgiem zastosowań Ada wpływa w sposób znaczący na konstruowanie oprogramowania systemów wojskowych. Zmieniają się metody doboru i szkolenia programistów, uaktualnieniu ulegają podstawowe metody projektowania, a w niektórych wypadkach tworzy się je wręcz od nowa. Język zaczyna wpływać nawet na rozwiązania sprzętowe.

Wszystkie te zmiany są bardziej zrozumiałe, jeśli weźmie się pod uwagę bieżące zastosowania Ady. Początkowo bowiem Ada była stosowana do przetwarzania danych tylko w systemach, w których istniały jej kompilatory, tzn. na dużych komputerach i superminikomputerach. Zasięg aplikacji rozszerzono wraz z pojawianiem się kompilatorów skrośnych i systemów wykonawczych na komputery wbudowane.

Pierwsza wersja systemu wbudowanego o nazwie AFADTS (ang. *Advanced Field Artillery Tactical Data System)*, ukończona przed dwoma laty przez firmę Magnavox, stanowi mieszaninę tradycyjnego przetwarzania danych i działania w czasie rzeczywistym. Stanowią go przenośne, mikrokomputerowe stacje graficzne do sterowania ogniem. Kod systemu liczy około 750 000 wierszy w Adzie.

Po rozszerzeniu przez Departament Obrony USA zaleceń w sprawie użycia Ady praktycznie do wszystkich programów w zastosowaniach krytycznych, staje się ona definitywnie językiem programowania wbudowanych systemów czasu rzeczywistego. Większość nowych kontraktów dla wojska, jak np. ulepszenie wozu bojowego Bradley, rakiet Lance, projekt stacji wojskowych i systemu bezpieczeństwa powietrznego realizuje się w Adzie. Istnieje też szansa realizowania w Adzie całych systemów uzbrojenia, np. wszyscy wykonawcy uczestniczący w projekcie nowoczesnego myśliwca taktycznego wykorzystują Adę, poczynając od programowania urządzeń przeciwdziałania elektronicznego, a kończąc na sterowaniu silnikiem.

Z pewnością głównym czynnikiem stymulującym rozwój języka jest opieka Departamentu Obrony, ale nie tylko. Wydaje się, że firmy, które zdobyły doświadczenie w Adzie, pracując na potrzeby armii USA zaczynają stosować ten język również gdzie indziej. Wszystko wskazuje na to, że samolot pasażerski Boeing 7J7 i system automatyzacji dla Federalnej Administracji Lotnictwa będą oparte całkowicie na Adzie.

Wiele firm jest gotowych do stosowania Ady, kiedy zajdzie potrzeba, ale nie za wszelką cenę. Po wieloletnim inwestowaniu, w inne języki wymagane przez Departament Obrony, nie można obecnie odrzucić żadnego oprogramowania wymagającego pielęgnacji. W rezultacie te firmy stosują Adę w pojedynczych programach. Jeśli kolejne zabiegi pielęgnacyjne obejmują nie więcej niż 30% projektu, to nie ma potrzeby dokonywania konwersji na Adę. Z tego powodu starsze języki programowania z pewnością będą jeszcze długo wykorzystywane.

Podejście konserwatywne niekoniecznie jednak oznacza sceptycyzm w stosunku do Ady. Projekty pilotowe w firmie Hughes Software dowiodły, że dzięki pewnym koncepcjom inżynierii oprogramowania zaimplementowanym w Adzie znacznie wzrasta efektywność programistów. Wydłużony o 10% czas kodowania rekompensuje z nawiązką zyski w czasie testowania.

Firma Magnavox stwierdziła, że projekt AFATDS nie mógłby powstać w wymaganym czasie przy wykorzystaniu innego języka niż Ada. W tych pracach wykorzystano podejście obiektowe oparte na systemie wspomagania tworzenia oprogramowania w Adzie firmy Rational.

ZNACZENIE SZKOLEŃ I NARZĘDZI

Większość przedsiębiorstw przyznaje, że żaden inny język nie miał takiego wpływu na dobór kadry, sposób szkolenia programistów i metody projektowe. Cały personel firmy Magnavox przeszedł rozległy program szkolenia w Adzie. Kursy dla personelu kierowniczego trwały pięć tygodni, a dla inżynierów cztery. Oceniając je z perspektywy należy powiedzieć, że nauka języka była najmniej istotną ich częścią, nacisk należy kłaść na konstruowanie oprogramowania; w przeciwnym razie będą powstawać programy pisane w Adatranie (Adatran odnosi się do nieefektywnego programowania w Adzie, podobnie jak w Fortranie).

Dystrybutorzy oprogramowania wydają się również doceniać wagę szkolenia dla firm, które przestawiają się na Adę. Program szkolenia firmy Verdix, opracowany wspólnie z firmą Honeywell, zawiera kursy poświęcone systemowi Verdix, jak również oddzielnie dla kierowników projektowych i trzytygodniowe kursy nauki języka. Z pewnością, uzasadnieniem tak wyczerpującego programu jest wpływ wywierany przez Adę na proces projektowy. Ada ma wbudowane pewne mechanizmy inżynierii oprogramowania, jak np. przesłanianie informacji. O ile takie właściwości wpływają na tworzenie lepszego kodu, to na pewno są niezgodne z istniejącymi metodologiami projektowymi poszczególnych firm i programistów. Dlatego na przykład firma Hughes zmodyfikowała swoją dotychczasową metodologię, aby dostosować ją do języka.

Kompilatory Ady wymagają lepszych niż przeciętne narzędzi projektowo-uruchomieniowych, nie z powodu natury języka, ale ze względu na najbardziej skomplikowane zastosowania. Ponieważ zmienia się istota zastosowań stają się one coraz bardziej złożone, pojawiają się też coraz lepsze narzędzia. Obecnie istnieją trzy warstwy narzędzi zwykle towarzyszących Adzie. Pierwsza warstwa wynika z wymagań odnośnie obsługi bibliotek, kontroli wersji i procedur sterujących budowaniem modułów. W drugiej warstwie znajdują się narzędzia poprawiające jakość kodu, takie jak optymalizatory, profilery i programy uruchomieniowe postaci źródłowej. Warstwa trzecia zawiera narzędzia najwyższego poziomu, tj. programy analizy projektowej lub strukturalnej, wymagane przede wszystkim z powodu dużej wielkości programów aplikacyjnych. Coraz częściej dystrybutorzy kompilatorów dostarczają pełny zestaw narzędzi dwóch pierwszych warstw.

Jako pierwsza na rynku firma Rational próbuje zunifikować wszystkie trzy warstwy. Impulsem do rozpoczęcia tych prac było stwierdzenie, o czym przekonano się dopiero po stworzeniu kompilatora, że obliczeniowy model Ady różni się zupełnie od języków klasycznych. Nie istniały zatem odpowiednie narzędzia i nie było rzeczywistego sprzężenia z konwencjonalnymi narzędziami analizy projektowej i strukturalnej. W środowisku akademickim dojrzewały w tym czasie idee ukierunkowane na techniki obiektowe, co wydawało się spójne z podłożem Ady. Dysponując taką oceną sytuacji, firma Rational opracowała

w Adzie obszerny, obiektowy system projektowy, dający duże oszczędności czasu w budowaniu programu, zarówno na etapie projektu wstępnego, jak i późniejszej kompilacji przyrostowej.

Po skompletowaniu oprogramowania projektowego napotkano nowy problem. Słaba lokalność, duża zależność od wielozadaniowości i niektóre inne cechy opracowanych narzędzi przeczyły założeniom architektury konwencjonalnej opartej na pamięci podręcznej. W celu zapewnienia możliwie najlepszej wydajności w firmie Rational zaprojektowano własny sprzęt przeznaczony dla tworzonego oprogramowania.

Większość kierowników projektowych Ady jest zdumionych „zachłannością" kompilatorów języka na zasoby komputera. O skali problemu świadczy przykład firmy Magna vox, która do wytworzenia 800 000 wierszy kodu potrzebowała następującego sprzętu: VAX 8800, VAX 8650, dwa komputery VAX 8600 i czternaście dysków o pojemności 460 MB. Naturalnie, taki głód zasobów zwraca uwagę wytwórców sprzętu, tym bardziej że społeczność Ady potrzebuje szybszej kompilacji, a właściwie zaimplementowana wieloprocesorowa wersja Ady powinna zapewnić pożądaną efektywność. Firma Encore przeprowadziła testy początkowej, jednostrumieniowej wersji kompilatora firmy Verdix na systemie Encore 120, które trwały 3 godziny i 15 minut, w porównaniu z 4 godzinami i 20 minutami w systemie wieloprocesorowym najgroźniejszych konkurentów.

Firma Encore planuje kontynuację prac nad Adą w środowisku wieloprocesorowym. Obecnie modyfikuje się system wykonawczy w celu osiągnięcia przezroczystej wieloprocesorowości, ponieważ wersja jednostrumieniowa nie wykorzystuje w odpowiednim stopniu architektury sprzętowej. W firmie panuje przekonanie, że połączenie nowego, współbieżnego kompilatora z komputerem Encore 320 (opartym na procesorze 32332) pozwoli zredukować czas potrzebny na wykonanie tekstów do około półtorej godziny.

Równie entuzjastyczne nastroje panują w firmie Masscomp, gdzie trwają prace nad rozszerzeniem własnej wersji kompilatora Verdix, wykorzystującego dwuprocesorowe stanowisko robocze. Wielu obserwatorów jest głęboko przekonanych o nieuchronnym i znacznym wzroście możliwości kompilatorów i sprzętu. Przedstawiciele firmy Telesoft twierdzą, że różnica jakości między kompilatorem pierwszej i drugiej generacji (ukończonym w 1986 r.) sięga rzędu wielkości.

SYSTEMY CZASU RZECZYWISTEGO

Nawet jeśli projektanci akceptują surowe wymagania metodologiczne Ady i stwarzają możliwości wykonywania coraz doskonalszych i wyrafinowanych narzędzi projektowych, to istnieje poważna przeszkoda na drodze do powszechnej akceptacji Ady – projektowanie systemów czasu rzeczywistego.

Firma Boeing zdefiniowała trzy klasy obiektów powietrznych. Przede wszystkim, istnieją systemy całkowicie cykliczne, w których zadania powtarzają się w tej samej kolejności, np. systemy naprowadzania pocisków. Następna klasa to systemy mieszane, o charakterze cyklicznym, ale z pojawiającym się obciążeniem asynchronicznym. Trzecia klasa charakteryzuje się wyłącznie zdarzeniami asynchronicznymi. Gdy projektowany system można zaliczyć do klasy drugiej lub trzeciej, projektanci muszą stawić czoła poważnym problemom.

Przede wszystkim, takie systemy nie mogą być zdeterminowane i ze swojej natury nie są podatne na testy. Nie można w nich tak, jak w systemie zdeterminowanym, określić a priori dokładnej sekwencji wykonania każdego modułu. Takie rozróżnienie w szczególności dotyczy przedsiębiorstw lotniczych pracujących na potrzeby rynku cywilnego i wojskowego. Federalna Administracja Lotnictwa (FAA) wymaga, aby programy wykorzystywane w lotnictwie cywilnym były zdeterminowane, natomiast Departament Obrony zainteresowany możliwościami modyfikacji oprogramowania nie czyni takich zastrzeżeń.

Członkowie zespołów projektujących asynchroniczne systemy czasu rzeczywistego w Adzie mają dwie możliwości: pisać program jako pojedyncze zadanie jawnie obsługujące szeregowanie zdarzeń asynchronicznych lub wykorzystać cechę wielozadaniowości języka. Co najmniej jedno duże przedsiębiorstwo pracujące na zamówienie armii USA wybrało pierwsze podejście. Powołując się na złe doświadczenia ze standardowymi systemami wykonawczymi (dostarczanymi przez wytwórców kompilatorów), w których nie można przełączać zadań w czasie kilku mikrosekund, jak wymaga się w zastosowaniach lotniczych, postanowiono nie wykorzystywać wielozadaniowości Ady w nowych projektach.

Dystrybutorzy kompilatorów zdają sobie sprawę z trudności pojawiających się w systemach czasu rzeczywistego. Przełączanie zadań w czasie kilku milisekund nie stanowi problemu przy przetwarzaniu informacji w środowisku superminikomputerów, ale jest nie do przyjęcia w systemach naprowadzania (ang. *guidence*). Nawet jeśli efektywność przełączania zadań poprawi się pięciokrotnie, to użytkownicy systemów lotniczych nadal będą unikać wielozadaniowości. Dzieje się tak, ponieważ wielozadaniowość przejawia cechy indeterminizmu, a w zagadnieniach lotniczych czasy odpowiedzi muszą być określone bardzo dokładnie.

Jednak nawet w tej dziedzinie przeprowadzane badania są obiecujące. Firma Boeing uważa, że kod niektórych kompilatorów można znacznie poprawić po wyeliminowaniu pewnych kontroli i uzgodnień. Firma Magnavox ma podobne spojrzenie na sprawę twierdząc, że wytwórcy kompilatorów przyjęli bardzo uproszczone podejście do wielozadaniowości tylko po to, aby uruchomić ten mechanizm. W celu osiągnięcia rzeczywistych możliwości języka, należy w znacznym stopniu poprawić realizację wielozadaniowości i rodzajowości w istniejących kompilatorach.

Dużo poważniejszą sprawą jest jednak rozdźwięk między obecnym stosowaniem Ady jako języka przede wszystkim jednoprocesorowego a praktyką projektowania systemów wieloprocesorowych. Przykładowo, architektura Pave Pillar dla systemów lotniczych tolerujących uszkodzenia składa się z zestawów dokładnie identycznych procesorów „misyjnych" i sygnalizacyjnych. Taka architektura powinna być punktem wyjścia do jakichkolwiek rozważań o środowisku docelowym Ady.

Można powiedzieć, że Ada sprawia wszystkim poważne kłopoty wskutek umieszczenia wielozadaniowości w systemie wykonawczym. Przenosi to odpowiedzialność za szeregowanie zadań czasu rzeczywistego na projektantów kompilatora, którzy nie są z tym zupełnie zaznajomieni. Zagadnienie jest szczególnie ważne w wieloprocesorowych systemach czasu rzeczywistego, na czym mało kto się zna.

• • •

Tego rodzaju wyzwania powodują konieczność uwzględnienia Ady przy projektowaniu architektury sprzętu. Ponadto, coraz intensywniejsze badania prowadzi się nad wykorzystaniem Ady w dużych, rozproszonych sieciach. Skutkiem tej sytuacji będzie bez wątpienia nowe spojrzenie na systemy wykonawcze i sprzęt. Jednakże nie należy oczekiwać zbyt daleko idących rozwiązań, podobnych jak w przypadku Lispu. Projektanci sprzętu nie mogą zaimplementować w krzemie algorytmów, które nie są jeszcze dojrzałe. Lisp ma bardzo dobrze zdefiniowane jądro, które można umieścić w sprzęcie, natomiast większość szczegółów wewnętrznych Ady nie jest tak dobrze zrozumiana.

Mimo tych wszystkich problemów, zespoły które podjęły znaczące wysiłki w celu stosowania Ady – przez szkolenia, kompletowanie oprogramowania narzędziowego i adaptację metodologii – wydają się być przekonane, że poczynione inwestycje są opłacalne.

Oprac.M.KUC
na podstawie Computer Design

Słownik pojęć i terminów

dokończenie z III okładki

WZMACNIANIE OBRAZU *(image enhancement)* – proces ulepszania wyglądu OBRAZU CYFROWEGO za pomocą takich technik, jak ROZCIĄGANIE KONTRASTU, WZMACNIANIE KRAWĘDZI, WYGŁADZANIE, WYOSTRZANIE I MANIPULOWANIE SKALĄ JASNOŚCI.

WZORZEC, OBRAZ (2) (*pattern*) – regularność, która może być wykorzystywana do KLASYFIKOWANIA OBIEKTÓW, zdarzeń lub innych danych.

WOJCIECH MOKRZYCKI

PRZEDSIĘBIORSTWO EKSPORTU USŁUG TECHNICZNYCH
„EKSBUD – KIELCE"

25-363 Kielce, ul. Wesoła 51

ZATRUDNI

w ośrodku informatycznym, do pracy na nowoczesnym, oryginalnym sprzęcie IBM, następujących pracowników:

- *informatyków*
- *elektroników*
- *projektantów systemów informatycznych*

Przedsiębiorstwo oferuje ciekawą, atrakcyjną pracę, kontrakty zagraniczne, konkurencyjne płace, mieszkania.

Oferty prosimy kierować do Działu Kadr, tel. 411-31 wewn. 236 lub Zespołu ds. Informatyki, tel. 31-28-33 wewn. 269.

EO/343/89

PC-ARK
SPÓŁKA Z O.O.
ul. Jaracza 3
00–378 Warszawa

firma, która posiada duże doświadczenie w zakresie automatyki przemysłowej, oferuje systemy:

* P – PROGRAMMABLE
* L – LOGIC
* C – CONTROLLERS

do sterowania procesami produkcyjnymi
od:
pojedynczych maszyn i stanowisk technologicznych
do:
kompleksowej automatyzacji linii i wydziałów produkcyjnych
możliwe do zastosowania we wszystkich gałęziach przemysłu.
Sieć punktów serwisowych na terenie całego kraju prowadzi:

* instalacje
* okresowe przeglądy konserwacyjne
* usługi gwarancyjne i pogwarancyjne

tel. 26 09 09, 26 27 94, 26 41 18
teleks 816962 pc pl

EO/127/89

CZAS TO PIENIĄDZ!

Błyskawicznie wykonają obliczenia systemy mikrokomputerowe **COMPLEX 386** zgodny z **COMPAQ 386**

- 16/20 MHz, 80386 32 Bit, SCRAM 1–2 MB, max. 4 MB, 65 ns, podstawka pod koprocesor 80387

COMPLEX 386 CACHE

- 20/25 MHz, 80386 32 Bit, DRAM Simm 1/4 MB, max. DRAM Dil 8 MB, DRAM Simm 16 MB, Cache 32 kB/35 ns, Austek 38152 kontroler, podstawka pod koprocesor 80387, Weitek 1167

oferowane jako:

- izolowane stanowiska prac obliczeniowych, z instalacją systemu operacyjnego DOS
- systemy wielodostępne z terminalami, z instalacją systemu operacyjnego XENIX V 386

Przyspieszy obliczenia PC XT/AT/386 system transputerowy **COMLINK 01**

- 1–4 transputery T414 lub T800, 1 lub 4 MB RAM dla każdego transputera
- oprogramowanie: kompilatory Fortran, Pascal, C

Natychmiast wydrukuje wyniki obliczeń drukarka japońskiej firmy **OKIDATA**

- mozaikowa 10" do 300 zn./s; 15" do 450 zn./s; „heavy duty" do 350 zn./s
- laserowa 6 stron/min

Prosimy pamiętać:
Czas to pieniądz!
Już czas podjąć decyzję!

Oferując ww sprzęt

Przedsiębiorstwo Zagraniczne KOMPLEX EFC
Zakład Elektroniki, 61-706 Poznań, ul. Libelta 6
tel. 53-17-93, teleks 414279

- udzieli szczegółowych informacji
- zapewni fachowe doradztwo, będące sumą doświadczeń sześcioletniej działalności na polskim rynku komputerowym
- opracuje koncepcje komputeryzacji
- wdroży system do eksploatacji

Ponadto PZ KOMPLEX EFC

- oferuje systemy wielodostępne w oparciu o PC COMPLEX AT, z instalacją systemu operacyjnego XENIX V 286
- prowadzi serwis autoryzowany sprzętu mikrokomputerowego na rzecz firmy HANSAPOOL, Handelsges mbH, An der Alster 6, 2000 Hamburg 1, RFN.

EO/354/89

Turbo Pascal 4.0

Nowa wersja kompilatora Turbo Pascala firmy Borland International ma wiele nowych własności, przede wszystkim pozwala podzielić program na rozłącznie kompilowane moduły o wielkości 64 KB, zwane jednostkami (ang. *units*). Kompilator pracuje na mikrokomputerach IBM PC, PC XT, PC AT, PS 2 lub zgodnych, pod nadzorem systemu operacyjnego MS-DOS wersji 2.0 lub wyższej. Mikrokomputer powinien posiadać nie mniej niż 256 KB pamięci RAM dla wersji liniowej (plik *TPC.EXE*) lub 384 KB przy korzystaniu ze zintegrowanego środowiska kompilatora sterowanego przez menu.

Nowa wersja charakteryzuje się ponadto możliwością tworzenia bibliotek skompilowanych jednostek (modułów), istnieniem narzędzi do zarządzania większym projektem, możliwością kompilacji warunkowej, dwoma trybami wyznaczania wartości wyrażeń logicznych, nowymi typami danych i możliwością korzystania z procedur zewnętrznych napisanych np. w języku asemblera. Nie należy również zapomnieć o dołączanych standardowo programach usługowych i licznych programach demonstracyjnych, m. in. prostego arkusza kalkulacyjnego *MicroCalc*.

Wersja 4.0 nie jest bezpośrednio zgodna z poprzednimi, dlatego w dotychczasowych programach należy dokonać pewnych konwersji. Do tego służy program *Upgrade*. W porównaniu z wersją 3.0 – obecna nie ma możliwości definiowania procedur nakładkowych.

WYKORZYSTANIE JEDNOSTEK

Jednostka jest zbiorem stałych, zmiennych, procedur i funkcji. Każda jednostka składa się z dwóch części: część sprzęgająca *INTERFACE*, zawiera informacje widoczne w innych programach wykorzystujących daną jednostkę, część implementacyjna, *IMPLEMENTATION*, zawiera informacje prywatne dla jednostki i niewidoczne na zewnątrz, ale wykorzystane przez część sprzęgającą. Jednostki nie są plikami włączalnymi (ang. *include*), natomiast są skompilowane i przechowywane w postaci maszynowej. Skompilowana jednostka Turbo Pascala jest zapisywana na pliku o rozszerzeniu *TPU* i jest w pewnym stopniu analogiczna do plików wynikowych *OBJ*. Do zarządzania biblioteką skompilowanych jednostek służy program usługowy *TPUMOVER*. Za pomocą tego programu można do biblioteki znajdującej się na pliku *TURBO.TPL* dodawać nowe jednostki lub usuwać niewykorzystywane (także standardowe).

Biblioteka dostarczana z kompilatorem zawiera siedem standardowych jednostek: *system*, *printer*, *crt*, *dos*, *graph*, *turbo3* i *graph3*. Jednostka *system* zawiera standardowe i wbudowane funkcje i procedury Turbo Pascala, jak również kilka dodatkowych jednostek. Każdy program korzysta z tej jednostki, natomiast jednostka *system* nie korzysta z żadnej innej jednostki. Jednostka *printer* definiuje zmienną *Lst* i związuje z programem obsługi drukarki, dzięki czemu procedury *write* i *writeln* wyprowadzają informacje na drukarkę. Jednostka ta korzysta z jednostki *crt*. Jednostka *crt* jest pakietem we-wy specyficznym dla mikrokomputerów PC udostępniającym procedury i funkcje obsługi klawiatury i ekranu. Jednostka ta jest głównym sprzężeniem we-wy dla tych urządzeń i może zostać wymieniona na jednostkę napisaną przez użytkownika, obsługującą niezgodne ze standardem PC klawiaturę i ekran. Jednostka ta nie wykorzystuje innych jednostek standardowych. Jednostka *dos* definiuje procedury i funkcje odpowiadające wywołaniom DOS-a, np. *GetTime* i *SetTime*. Za pomocą zaimplementowanej w tej jednostce procedury *Intr* można wykonać dowolne przerwanie programowe pod warunkiem, że nie korzysta ono z rejestru wskaźnika stosu *SP* i rejestru segmentu stosu *SS*. Z natury rzeczy, takie samo ograniczenie dotyczy procedury *MsDos*, istniejącej już w wersji 3.0 kompilatora.

Jednostka o nazwie *graph* składa się z ponad 50 funkcji i procedur mogących współpracować z siedmioma adapterami graficznymi (*CGA*, *MCGA*, *EGA*, *VGA*, *Hercules*, *AT&T* i 3270). Tak wiele różnych rozdzielczości jest obsługiwanych dzięki ustaleniu współrzędnych lewego górnego rogu ekranu na (1,1), zamiast skalowania danych pochodzących z wirtualnego ekranu na współrzędne danego wyświetlacza. Procedury i funkcje jednostki można podzielić na kilka grup: udostępniające parametry wyświetlacza, takie jak szerokość, wysokość (w pikselach) lub współczynnik kształtu, podstawowe procedury graficzne zaznaczania punktu, rysowania odcinka lub ścierania obrazu, bardziej złożone służące do rysowania łuków i okręgów, i wreszcie procedury wykonujące skomplikowane zadania, takie jak narysowanie wykresu słupkowego, kołowego lub zamalowanie wzorem obszaru ograniczonego wielokątem – niekoniecznie wypukłym. Rysunki można opisywać wykorzystując jeden z czterech krojów pisma. Trzy spośród nich składają się ze znaków kodowanych wektorowo. Tego rodzaju kroje pisma łatwiej przeskalować w wypadku zmian rozdzielczości. Jednostka *graph* odwołuje się do jednostki *crt*. Moduł *graph3* obsługuje procedury graficzne dostępne w wersji 3.0 i również wykorzystuje jednostkę *crt*.

Jednostka *turbo3* zawiera definicje procedur pozwalających utrzymać zgodność z wersją 3.0 kompilatora i dublujących procedury występujące w innych jednostkach, szczególnie w *crt*. Są to: *Kbd*, *CBreak*, *MemAvail*, *MaxAvail*, *LongFileSize*, *LongFilePos*, *LongSeek*, *IOResult*, *NormVideo*, *LowVideo* i *HighVideo*. Wykorzystane w wersji 3.0, zdefiniowane pierwotnie urządzenie *Kbd* do odczytywania bez echa znaków z klawiatury jest niedostępne w wersji 4.0. Zamiast tego można korzystać z procedury *Readkey* jednostki *crt*. Za pomocą funkcji KeyPressed można sprawdzić, czy naciśnięto klawisz. W wersji 3.0 błędnie zaimplementowano tę funkcję, co teraz poprawiono. W wersji 4.0 dla klawiszy funkcyjnych, kursora itd., pierwszym znakiem jest *NULL*, zamiast *ESC*. Zmienna *CheckBreak* w wersji 3.0 nazywała się *CBreak*. Funkcje *MemAvail* i *MaxAvail* podają wielkość dostępnej i maksymalnej pamięci wyrażoną w bajtach, poprzednio – wyrażoną w paragrafach. W związku z wprowadzeniem nowych typów, wartości udostępniane przez funkcje *LongFileSize*, *LongFilePos* i *LongSeek* są teraz typu *LongInt*. Funkcja *IOResult* udostępnia standardowe kody błędów systemu MS-DOS. Procedury *NormVideo*, *LowVideo* i *HighVideo* pozwalają odpowiednio odwzorować kolory o większej intensywności na kolory mniej intensywne i odwrotnie, jak również – przywrócić stan standardowy. Jednostka *Turbo3* wykorzystuje jednostkę *crt*.

NARZĘDZIA ZARZĄDZANIA PROJEKTEM

Pakiet zawiera programy usługowe *MAKE*, *TOUCH* i *GREP*; narzędzia znane wielu programistom języka C. Program *MAKE* jest samodzielną procedurą wywołującą kompilatory, asemblery i inne programy generujące pliki wynikowe, wykorzystywane przez kompilator Turbo Pascala do połączenia w jeden wykonywalny plik. *TOUCH* to program uaktualniający datę i czas stworzenia pliku, tak aby miał datę późniejszą od wszystkich plików zależnych od niego. Jest to potrzebne po to, aby wywołanie programu *MAKE* powodowało przekompilowanie całości. Program *GREP* służy do szukania wzorców w jednym lub większej liczbie plików.

Zintegrowane środowisko wersji 4.0 zawiera opcje menu służące zarządzaniu projektem: *File*, *Edit*, *Run*, *Compile* i *Option*. Na przykład, opcja *Pick* z menu *File* pozwala wybrać jeden z ośmiu ostatnio wykorzystywanych programów plików do redagowania i ustawia kursor w pozycji, w jakiej był w chwili zapisania. Menu *Compile* zawiera opcje: *Compile*, *Make* i *Build*. Opcja *Compile* kompiluje plik aktualnie redagowany. Opcja *Make* sprawdza, czy jednostki wchodzące w skład programu zostały zmienione od czasu ostatniej kompilacji; znalezienie zmian powoduje rekompilację jednostki. Inaczej mówiąc, *Make* zapewnia, że program wykorzystuje najnowsze wersje jednostek. Opcja *Build* kompiluje wszystkie jednostki składające się na program, bez względu na ich status.

Opcje kompilatora mogą być ustawione w programie źródłowym jako dyrektywy kompilatora:

$dyrektywa+ powoduje ustawienie opcji
$dyrektywa– wyłącza opcję.

Można je również wybrać w menu *Compile* środowiska zintegrowanego lub – jako parametry polecenia kompilacji wersji liniowej kompilatora ($X+ włącza, $X wyłącza opcję)...

KOMPILACJA WARUNKOWA

Turbo Pascal wersji 4.0 umożliwia kompilację warunkową części programu, co można wykorzystywać przy jego uruchamianiu. Następujące dyrektywy definiują kompilację warunkową:

$DEFINE symbol definiuje symbol do wykorzystania przez inne dyrektywy kompilatora,
$UNDEF symbol usuwa definicję,
$IFDEF symbol kompiluje występujący niżej kod, jeśli podany symbol jest zdefiniowany,
$IFNDF symbol kompiluje, jeśli symbol nie jest zdefiniowany,
$IFOPT dyrektywa+ kompiluje kod, jeśli dyrektywa jest włączona,
$IFOPT dyrektywa– kompiluje kod, jeśli dyrektywa jest wyłączona.
Dyrektywa *$ELSE* działa tak samo, jak *ELSE* w instrukcji *IF...THEN...ELSE* i może wystąpić po dowolnej dyrektywie *$IFxxx*. Dyrektywa *$ENDIF* wskazuje koniec sekcji *$IFxxx* lub *$ELSE*.

Powyższe uwagi dotyczą dyrektyw umieszczanych w kodzie źródłowym, ale te same wyniki można osiągnąć za pomocą parametrów polecenia lub opcji zintegrowanego środowiska. Można również ustawić symbol lub opcję kompilatora i wykorzystując je jako wyrażenie logiczne sterujące kompilacją segmentu kodu. Wyrażenia logiczne mogą być obliczane dwoma sposobami, w zależności od dyrektywy *$B*. Włączenie powoduje obliczanie całego wyrażenia i działanie na wyniku, a po wyłączeniu dyrektywy wyrażenie jest obliczane tylko do miejsca, w którym znany jest wynik (prawda lub fałsz).

Oprócz tego programista ma do dyspozycji dyrektywy włączające i wyłączające sprawdzanie indeksów tablic (*$R*), przepełnienia stosu (*$S*) operacji we-wy (*$I*), zmiennych napisowych przesyłanych jako parametry (*$V*), generowanie pliku *TPM* wykorzystywanego do znalezienia błędów wykonania (*$T*), generowanie informacji do uruchamiania programu (*$D*) lub generowanie programowe lub sprzętowe (przez koprocesor) operacji zmiennoprzecinkowych (*$N*).

DODATKOWE CECHY

Poprzednie wersje Turbo Pascala miały swoje odmiany przeznaczone do pracy nad nadzorem systemu PC-DOS lub MS-DOS. Wersja 4.0 jest wspólna dla obydwu systemów operacyjnych. Jednakże w programach powinno się unikać jednostek lub opcji specyficznych dla mikrokomputerów firmy IBM.

W wersji 4.0 zaimplementowano nowe typy danych, mianowicie:

Longint – 4-bajtowe liczby całkowite ze znakiem,
Byte – 1-bajtowe liczby całkowite bez znaku,
Shortint – 1-bajtowe liczby całkowite ze znakiem,
Word – 2-bajtowe liczby całkowite bez znaku,
Single – 4-bajtowe liczby zmiennoprzecinkowe (7 cyfr znaczących),
Double – 8-bajtowe liczby zmiennoprzecinkowe (15 cyfr znaczących),
Extended – 10-bajtowe liczby zmiennoprzecinkowe (19 cyfr znaczących),
Comp – 8-bajtowe liczby całkowite z zakresu od $-2^{63}+1$ do $2^{63}-1$.

Ostatnie cztery typy danych odpowiadają normie IEEE-754, ale wymagają obecności ko-procesora arytmetycznego; Turbo Pascal nie ma trybu jego emulacji.

Fragmenty kodu maszynowego można dołączać do programu za pomocą instrukcji lub dyrektywy *inline*. Instrukcja *inline* ma działanie podobne do wersji 3.0, jednak większość segmentów należy zmodyfikować ze względu na istniejące różnice w schemacie wywołań (konieczność zachowania rejestrów *BP*, *SP*, *SS* i *DS*). Możliwe jest również stosowanie dyrektywy *inline*, będącej w rzeczywistości makrodefinicją, pozwalającej programować procedury lub funkcje. Wywołanie takiej procedury równoważne jest rozwinięciu makrodefinicji.

W wersji 3.0 procedury obsługi przerwań trzeba było kodować w języku asemblera. Obecnie można je pisać jako procedury Pascala o typie *interrupt*.

ZGODNOŚĆ KOLEJNYCH WERSJI

Poprzednie wersje Turbo Pascala (2.0 i 3.0) wykazały całkowitą zgodność między sobą. Dla wersji 4.0 nie jest to prawdą. Z tego względu narzędzia dołączone do pakietu są bardzo pożyteczne lub wręcz niezbędne. Dwie jednostki biblioteczne (*graph*3 i *turbo*3) służą zachowaniu zgodności z poprzednią wersją, a program *Upgrade* przekształca programy źródłowe wersji 3.0 na postać wymaganą w wersji 4.0. Poprawki są wstawiane w tekst programu źródłowego, jak również w plikach dołączonych dyrektywą *Include*. Dodatkowo można otrzymać plik opisujący dokonane zmiany, wraz z obszernym komentarzem (opcja *J*) lub podzielić program na jednostki (opcja *U*).

W nowej wersji Turbo Pascala nie ma możliwości dzielenia programu na nakładki. Istnieje możliwość przekształcenia programu wykorzystującego nakładki na równoważny program podzielony na jednostki (nie przekraczające 64 KB). W tym celu należy wykorzystać opcję/*U* dla *Upgrade* i w odpowiednich miejscach programu umieścić dyrektywę.

{. U nazwa jednostki}

Jednakże, jeśli korzystano z nakładek w wypadku, gdy program nie mieścił się w dostępnej pamięci, to należy go zmodyfikować. Program główny i wszystkie jednostki muszą mieścić się w pamięci jednocześnie. Firma twierdzi, że inteligentne zarządzanie nakładkami zostanie włączone do kompilatora w najbliższym czasie. Byłoby to korzystne dla użytkowników, ponieważ struktura nakładkowa wersji 3.0 była bardzo efektywna.

Zmieniono również kolory wyświetlanych znaków z żółtego na biały, tło pozostało to samo – czarne. Standardowy układ kolorów można zmienić za pomocą pierwotnie zdefiniowanych procedur *SetColor* i *SetBkColor*. W zależności od zainstalowanego programu obsługi i trybu graficznego można wybierać spośród 16 kolorów.

TESTY PORÓWNAWCZE

Wersje 3.0 i 4.0 porównano za pomocą standardowego zestawu testów miesięcznika Byte. Niewielki program pomocniczy *RUN.PAS* służył do uruchomienia testów i pomiaru czasów wykonania. *RUN.PAS* mierzył całkowity czas wykonania, włączając w to czas potrzebny na załadowanie przez system operacyjny programu z dyskietki. Wszystkie testy napisano i wykonano w Turbo Pascalu 3.02a, a następnie przekształcono programem *Upgrade* na wersję 4.0 i ponownie wykonywano.

Tabela 1. Testy porównawcze

Nazwa testu	Czas wykonania (w sekundach)		Wielkość plików wykonywalnych (w bajtach)	
	3.02a	4.0	3.02a	4.0
CALC	32.68	23.51	11 635	4432
FLOAT	66.51	54.54	11 543	3536
SIEVE	16.37	14.61	11 713	2720
TRANS	80.08	79.21	11 544	2912
BTRAS	7.63	6.05	11 560	2928
LINETEST	21.09	5.98	11 533	8704
HEAPTEST	0.93	0.66	11 835	3104

Tabela 2. Testy porównawcze

Numer wersji	Czas wykonania		Wielkość plików wykonywalnych	
	CALC	*FLOAT*	*CALC*	*FLOAT*
3.02a	7.08	2.97	10 282	10 194
4.0 (4 bajty)	2.31	2.09	4704	2608
4.0 (8 bajtów)	2.64	2.09	4704	2608
4.0 (10 bajtów)	2.69	2.09	4720	2608

Tabela 1 zawiera zestawienie czasów wykonania i wielkości programów dla porównywanych wersji Turbo Pascala. Test *CALC.PAS* wykazuje, że mnożenie i dzielenie 6-bajtowych liczb zmiennoprzecinkowych jest o 39% szybsze w nowej wersji kompilatora. Program *FLOAT.PAS* testujący transcendentne funkcje biblioteczne również wykorzystuje 6-bajtowe liczby zmiennoprzecinkowe. Wersja 4.0 wykazuje przewagę 22%. Program *SIEVE. .PAS* testujący efektywność działań na tablicach wykazał tylko 12% przewagę nowej wersji. Program *TRANS.PAS* wykonuje kopiowanie pliku po jednym znaku, natomiast *BTRANS.PAS* wykonuje operację blokową (blok = 128 bajtów). Dla pierwszego testu różnica jest praktycznie nieistotna, tylko 1%, ale dla drugiego różnica w szybkości wykonywanych operacji jest znaczna, bo już 26%.

Procedura kreślenia odcinków *LINETEST. .PAS* wskazuje, gdzie istnieje naprawdę znacząca różnica między testowanymi wersjami kompilatora. W tym wypadku różnica w szybkości jest doprawdy imponująca, 253% na korzyść nowej wersji, przy czym procedura wymagała pewnych poprawek. Przy pierwszym wykonaniu, na ekranie nie było większości odcinków. Problem leżał w piątym parametrze procedury *DRAW*. Turbo Pascal w wersji 3.02a wykorzystywał tylko 2 mniej znaczące bity parametru określającego kolor, natomiast wersja obecna wykorzystuje wszystkie i nakłada na programistę obowiązek sprawdzenia zakresu argumentów. W tabeli 1 podano czas wykonania dla zmodyfikowanej procedury. W obydwu wypadkach wielkość wykonywalnego kodu była znacznie mniejsza. *HEAP-TEST.PAS* inicjalizuje i zwalnia wskaźniki, wskazujące na obszar 9999 bajtów. Różnica czasów wykonań wynosi ok. 41%.

Omawiane programy testowe są niewielkie i przeprowadzenie pomiarów czasu kompilacji nie byłoby łatwe. Dlatego przeprowadzono je na większym programie – skrośnym makroasemblerze na mikroprocesor 8085, nieznacznie zmodyfikowanym do kompilacji dla wersji 4.0. Program źródłowy wersji 3.02a miał objętość 94 140 bajtów, a w wersji 4.0 – 94 301 bajtów. Pewną część czasu kompilacji zajmowały operacje dyskowe wywołane istnieniem dyrektyw *INCLUDE*. W czasie tego testu kompilacja przeprowadzona w środowisku interakcyjnym wersji 4.0 była blisko dwa razy szybsza niż dla wersji starszej (28 s i 58 s). Pomiary wykonano z dokładnością do jednej sekundy. Obydwie kompilacje wykonano również na IBM PS/2 Model 50, gdzie czasy spadły odpowiednio do 14 s i 6 s. Wersja liniowa Turbo Pascala skompilowała program makroasemblera w ciągu 38,1 s, ale nie można dokonywać tak bezpośrednich porównań. Kompilację w tym wypadku uruchomiono programem *RUN.PAS* i podany czas obejmuje ładowanie kompilatora i samą kompilację.

W tabeli 2 porównano czasy wykonania i wielkość plików testowych *CALC.PAS* i *FLOAT.PAS* wykorzystujących koprocesor 8087. Dla wersji 4.0 wykorzystano dostępne typy danych 4-, 6- i 10-bajtowe. Dla nowej wersji Turbo Pascala test *CALC.PAS* wykonywał się od 163% (dla 10-bajtowych liczb zmiennoprzecinkowych) do 206% szybciej (dla 4-bajtowych). Test *FLOAT.PAS* wykazuje 42% przewagę wersji 4.0, niezależnie od wykorzystywanego typu danych. Zmiana wielkości wykonywalnego pliku jest znaczna.

Opracował
M. KUC
na podstawie Byte

Rocznica projektów standardów OSI

W Waszyngtonie odbyło się w marcu 1988 roku spotkanie podkomitetu normalizacyjnego *Open System Interconnection* (wzajemne połączenie systemów otwartych), którego wiele grup roboczych obchodzi dziesiątą rocznicę działalności. Międzynarodowa Organizacja Normalizacyjna ISO podjęła problematykę OSI w 1978 roku w odpowiedzi na decyzję podjętą w 1977 roku na plenarnym spotkaniu Komitetu Technicznego nr 97. Inauguracyjne posiedzenie utworzonego w tym celu podkomitetu 16 (przemianowanego później na 21) odbyło się na przełomie lutego i marca 1978 roku, również w Waszyngtonie, z udziałem garstki delegatów z kilku krajów. W posiedzeniu rocznicowym wzięło udział 400 delegatów z 22 krajów.

Podkomitet 16 zajmował się normami dolnej warstwy do połączeń między komputerami, natomiast podkomitet 21 opracował 90 norm międzynarodowych wyższych warstw. Na początku 1988 roku przygotowano do zatwierdzenia 88 nowych dokumentów, a 47 innych osiągnęło poziom propozycji wstępnych. Ponadto, Międzynarodowa Komisja Elektrotechniczna (*International Electrotechnical Commission* – IEC) opracowała własne normy, które uzupełniają zalecenia ISO OSI. W listopadzie 1987 roku obie te organizacje utworzyły wspólny komitet techniczny (*Joint Technical Committee* – –JTC1), który ma koordynować wszystkie projekty norm informatycznych opatrzonych oznaczeniem ISO/IEC/JTC1.

Równolegle Międzynarodowy Komitet Doradczy Telegrafii i Telefonii (CCITT) sfinalizował w latach 1985–1988 uzgodnienia odnośnie protokołów dla nowych usług komunikacyjnych, które miały być opublikowane w 1988 roku jako zatwierdzone zalecenia. Podkomitety ISO mają dodatkowe zadanie koordynowania standardów OSI z projektami zaleceń CCITT, co czyni je normami telekomunikacyjnymi do zastosowań międzynarodowych.

Zasadnicze punkty norm opracowanych przez podkomitety OSI stają się podstawą specjalnych norm dla różnych zastosowań opracowywanych przez inne międzynarodowe i krajowe organizacje normalizacyjne.

Na targach CeBIT '88 w Hannowerze firmy Siemens, Bull (Honeywell), ICL i Olivetti zademonstrowały wymianę kompletnych dokumentów biurowych pomiędzy ich systemami komputerowymi opierając się na normach OSI. Podobny pokaz był przygotowany na konferencji w Baltimore w czerwcu 1988 roku. Tak więc, sterowanie komunikatami i wymiana plików pomiędzy systemami komputerowymi różnych firm stały się rzeczywistością dzięki dekadzie intensywnego rozwoju norm międzynarodowych dotyczących wzajemnych połączeń komputerów. Wynikiem tych dziesięcioletnich wysiłków jest też pojawienie się wyrobów opartych na tych normach, co wpłynęło na strategię zmuszonych do współpracy dostawców systemów komputerowych. Pojawiły się nowe klasy zastosowań opartych na wzajemnym połączeniu systemów komputerowych.

Prognoza rozwoju drukarek

W ciągu 5 lat, między 1986 a 1991 rokiem, wzrost sprzedaży drukarek w Europie ma wynieść 27% (z 7,74 do 9,82 mld dolarów). W 1986 roku dominowały drukarki uderzeniowe, które stanowiły prawie 42% rynku, następnie kopiarki (ponad 40%), a dalej inteligentne kopiarki-drukarki (9,5%), powielarki (6,9%), drukarki kolorowe (1,2%) i fascimile (0,22%). Według prognoz na rok 1991 najszybszy wzrost sprzedaży (3,36 raza) wykażą drukarki kolorowe i ich udział wzrośnie do 3,3% rynku (z 95 do 319 mln dolarów). Również ponad trzykrotnie wzrośnie sprzedaż urządzeń fascimile, choć udział ich wynosić będzie nadal tylko 0,56%. Nieco mniejszy (2,62 raza) ma być wzrost sprzedaży drukarek-kopiarek inteligentnych, w których przygotowywany jest wydruk całej strony. Udział ich wyniesie prawie 20%. Ostatnim działem, którego sprzedaż ma wzrosnąć w 1991 roku o 23%, tj. prawie do 3,9 mld dolarów, są zwykłe kopiarki. Ich udział w całym rynku drukarek trochę się obniży (do 39,3%), ale i tak znajdą się na pierwszej pozycji, natomiast drukarki uderzeniowe, których sprzedaż praktycznie utrzyma się na niezmienionym poziomie (3,2 mld dolarów), a udział obniży się do 33%, spadną na drugą pozycję. Wyraźnie zmniejszy się sprzedaż powielarek (o19% z 530 do 432 mln dolarów), a udział ich spadnie do 4,4%.

Oznacza to, że drukarki bezuderzeniowe (zarówno jedno – jak i wielobarwne) są najszybciej rozwijającym się działem techniki w Europie. W stosunku do USA, które mają o 1/3 mniej ludności niż Europa Zachodnia, wykorzystuje się tu ponad 40% mniej drukarek i o połowę mniej komputerów, lecz właśnie dlatego są tu większe możliwości wzrostu. Dotychczas opóźnienie techniczne pomiędzy Stanami Zjednoczonymi a Europą szacowano na 2 lata. Obecnie ma ono szanse zmaleć.

Przyczyny opóźniające wprowadzenie nowoczesnego sprzętu biurowego w Europie polegają na rozbiciu językowym i rynkowym. Pomimo to możliwości rozwoju nowych technik są znaczne, zwłaszcza w Wielkiej Brytanii, gdzie liczba komputerów osobistych na jednego mieszkańca jest największa na świecie.

Zastosowanie dysków optycznych

Według przedstawiciela firmy FileNet, rok 1988 był przełomowym, jeśli chodzi o przetwarzanie obrazów, które zostanie włączone do istniejących systemów przetwarzania danych i automatyzacji biur. Na tej samej stacji można będzie przeglądać obrazy z dysku optycznego, dane z komputera głównego i tekst np. z minikomputera, nie znając źródeł tych informacji ani systemów operacyjnych potrzebnych do ich uzyskania. Systemy takie są wykorzystywane w bankach, agencjach ubezpieczeniowych i innych instytucjach. Firma FileNet specjalizuje się w ich instalowaniu i przewiduje wzrost zastosowań w dziedzinie medycyny, prawa, produkcji i transportu. Jej oprogramowanie pozwala na przesyłanie obrazów przez sieć. Posługując się oprogramowaniem do przetwarzania obrazów na komputerach osobistych, użytkownicy mogą oglądać na swych monitorach obrazy przechowywane na odległych dyskach optycznych.

INTERSOFT

00-443 Warszawa, ul. Górnośląska 9/11
tel. 21-56-08, 28-67-94, teleks 817245

Szanowni Państwo!

Polecamy duży wybór dokumentacji dot. komputerów IBM oraz oprogramowania, m.in.:

Przewodnik programisty IBM	60 000 zł	Turbo Power Tools Plus (procedury do TP 4/5)	55 000 zł
Wprowadzenie do komputerów IBM	18 000 zł	Grafika TP v.4.0 i TC v.1.5	55 000 zł
Sidekick	23 000 zł	Clipper 86, kompil. do dBase III+	55 000 zł
Wstęp do grafiki (Basic, Turbo, Graphics)	45 000 zł	Clipper 87, kompil. do dBase III+	75 000 zł
Autocad v.2.17	50 000 zł	dBase III Poradnik encyklopedyczny	50 000 zł
Lotus 1–2–3 v.2.0	65 000 zł	dBase III+	50 000 zł
Framework IIp	70 000 zł	dBase III+, praca w sieci	20 000 zł
System operacyjny DOS 3.2	50 000 zł	Fox Base+	70 000 zł
System operacyjny DOS 3.3	85 000 zł	Programowanie w assemblerze	55 000 zł
Podręcznik programowania w GW-Basic	50 000 zł	Eureka	45 000 zł
Turbo Basic	70 000 zł	Poly-Windows	25 000 zł
Turbo „C" v.1.0	70 000 zł	Instrukcja obsługi PC 1512	35 000 zł
Turbo „C" v.1.5 t. 1 i 2	68 000 zł	Wordstar 2000	25 000 zł
Aztec „C"	60 000 zł	Modula 2 Logitech	48 000 zł
Zastosowanie języka C dla zaawansowanych	70 000 zł	Informix	90 000 zł
Programowanie w Turbo Pascal v.3.0	48 000 zł	OS-2, opis systemu	75 000 zł
Turbo Graphics (do TP v.3.0)	43 000 zł	Or-Cad	95 000 zł
Turbo Database Toolbox (do TP v.3.0)	24 000 zł	PC Tools De Luxe	15 000 zł
Turbo Pascal v.4.0	65 000 zł	Multi-Link v.4.0 (Lan Link 4.0)	50 000 zł
Turbo Pascal v.5.0	90 000 zł	Novell, podr. użytkownika i instalatora	150 000 zł

Do większości pozycji dołączamy dyskietki z przykładami (koszt nośnika nie jest ujęty w wyżej wymienionych cenach). Przy płatności czekiem lub gotówką (bądź przy sprzedaży za zaliczeniem pocztowym) udzielamy 10% zniżki.

Prowadzimy sprzedaż oraz skup wszelkiego sprzętu komputerowego oraz audio-video.

Do sprzedawanego przez nas sprzętu dodajemy bezpłatnie pakiety dokumentacji i oprogramowania o wartości zależnej od wielkości sprzedaży.

ZAPRASZAMY !

EO / 232 / 89

Edukacja informatyczna w Japonii

Trudno sobie wyobrazić, by bez odpowiedniego systemu powszechnej edukacji, także informatycznej, Japonia mogła utrzymywać swój wysoki poziom technicznego know-how oraz konkurować w świecie nowości technologicznych, zwłaszcza na rynku komputerowym. Powszechne wyobrażenia o tym kraju dość często stanowią duże uproszczenia rzeczywistości. Zalew rynku produktami przemysłu elektronicznego, zwłaszcza kalkulatorami, wyrobił w niejednym z nas przekonanie, że komputer (obecnie, a do niedawna – kalkulator) jest nieodłącznym towarzyszem życia Japończyków. Jest w tym na pewno dużo prawdy, gdyż komputery można spotkać w Japonii wszędzie tam, gdzie usprawniają działanie. Dużym osiągnięciem ostatnich lat stała się na przykład masowa produkcja podręcznych (tj. w postaci maszyny do pisania) mikroelektronicznych procesorów tekstów języka japońskiego, który ze zrozumiałych względów długo opierał się komputeryzacji.

Do nieprofesjonalnego użytku, Japończycy produkują natomiast tzw. rodzinne komputery (ang. *family computer*), przypominające zabawki i służące do gier i zabaw oraz do prostych obliczeń w pieleszach rodzinnych.

Nie ulega wątpliwości, że obecność komputerów wraz z dodatkowym wyposażeniem (kolorowymi monitorami, drukarkami itp.) w otoczeniu Japończyków od ich najmłodszych lat powoduje, iż przy pierwszym zetknięciu się z komputerami w szkole uczniowie mają już poza sobą pierwszą fascynację tymi urządzeniami. W tej sytuacji, za jeden z fenomenów oświaty w Japonii można uznać ugruntowaną i nie słabnącą pozycję sorobanu[1] – liczydła, będącego nadal jedną z podstawowych pomocy naukowych każdego ucznia w szkole podstawowej. Los kalkulatorów w japońskich szkołach jest podobny do ich sytuacji w naszych szkołach – nikt ich nie tępi, ale też nie zostały oficjalnie (przez umieszczenie w programie) dopuszczone do nauczania ani jako pomoc, ani jako cel zajęć.

A jakie miejsce w japońskich szkołach zajmuje edukacja komputerowa? Od samego początku trzeba rozgraniczyć szkoły wyższe od pozostałych szkół. W wypadku tych pierwszych można bowiem mówić o powszechnym nauczaniu informatyki, pewnym zaskoczeniem może być natomiast niemal całkowity brak regularnych zajęć z komputerami w trakcie obowiązkowego kształcenia.

By coś więcej powiedzieć o komputeryzacji japońskich szkół, warto najpierw przedstawić strukturę szkolnictwa w tym kraju[2]. Każdy młody Japończyk jest objęty dziewięcioletnim obowiązkowym (i bezpłatnym) nauczaniem – przez 6 pierwszych lat w szkole podstawowej (ang. *primary school, elementary school)* i 3 następne w szkole średniej (ang. *junior high school, middle school)*. Później może wybrać gimnazjum (ang. *high school*), do którego uczęszcza przez 3 lata, by następnie móc studiować w szkole wyższej (np. w uniwersytecie, akademii medycznej lub college'u). Ponad 90% uczniów kontynuuje naukę w gimnazjach i dla 90% uczniów dwunastoletnie wykształcenie nie jest wystarczające, uczą się więc dalej, z czego 35% w uniwersytetach. Jednym z nieodłącznych elementów obrazu szkolnictwa japońskiego są prywatne szkoły, w których nauka kosztuje olbrzymie kwoty (zwłaszcza w szkołach medycznych). Dzięki temu m. in. szkoły te jako zasobniejsze i bardziej elitarne, zapewniają swoim uczniom lepsze warunki nauczania, w szczególności są lepiej wyposażone w sprzęt komputerowy (por. tab.). Szkoły prywatne są również wyżej klasyfikowane zarówno w hierarchii społecznej, jak i na listach „rankingowych", co ma niemały wpływ na późniejsze koleje losów ich absolwentów. Całą prawdę o znaczeniu miejsc w tej klasyfikacji oddaje powszechny pogląd, iż chcąc umieścić kiedyś swoje dziecko w Tōdai (czyli w Uniwersytecie Tokijskim, zajmującym czołową pozycję wśród uniwersytetów), należy myśleć o tym już przy wyborze... przedszkola.

Do końca lat siedemdziesiątych programy nauczania w pierwszych trzech typach szkół nie zawierały żadnego przedmiotu, którego celem byłoby powszechne kształcenie informatyczne dzieci i młodzieży. Zajęcia z komputerami były nadobowiązkowe i odbywały się w godzinach pozalekcyjnych. Na niewielką skalę było również rozwinięte komputerowe wspomaganie nauczania. Potwierdzają to liczby zamieszczone w tabeli. Sytuacja ma ulec zmianie po wprowadzeniu w latach osiemdziesiątych nowych programów nauczania do wszystkich szkół. To ogromne przedsięwzięcie jest związane z olbrzymimi wydatkami na nowe podręczniki (udostępniane uczniom za darmo we wszystkich szkołach państwowych) oraz wyposażenie w sprzęt komputerowy.

Realizację tych planów rozpoczęto od gimnazjum. Do końca 1982 r. wprowadzono informatykę do programów nauczania w 2600 szkołach tego typu i w 36 z 47 prefektur utworzono centra informatyczne dla szkolnictwa. Z powodów finansowych, wyposażenie szkół w komputery rozpoczęto od gimnazjów technicznych i ekonomicznych, w których znalazło się 8000 komputerów, zatem średnio na jedną szkołę przypadło 4,1 komputera. W szczególności, 41% tych szkół otrzymało tylko jeden komputer, ale 3% szkół dostało aż po ponad 20 komputerów. Połowa wszystkich uczniów kończących gimnazja ma za sobą kontakt z komputerem. Chwalą to sobie zarówno szkoły wyższe, jak i przemysł przejmujący absolwentów tych szkół.

We wszystkich wyższych szkołach w Japonii nauczaniem elementów informatyki są objęci wszyscy studenci. Chociaż zakres i organizacja tych zajęć są równe dla różnych szkół, można w nich jednak wyróżnić wiele cech wspólnych, specyficznych dla edukacji informatycznej w wydaniu masowym. Pewnym ułatwieniem w prowadzeniu tych zajęć jest zapewne fakt, iż komputery wraz ze swoim dodatkowym wyposażeniem nie stanowią dla japońskich studentów żadnej nowości, która skupiałaby na sobie całą ich uwagę. Programy zajęć informatycznych i ich realizacja są ściśle związane z rodzajem uczelni i programami przedmiotów kierunkowych, są więc zatem różne dla różnych szkół. Wspólną zaś cechą tych zajęć jest mocne powiązanie ich treści aplikacyjnej z kierunkiem studiów. Zajęcia te odbywają się najczęściej podczas pierwszego roku studiów, gdyż ich celem jest m. in. przygotowanie studentów do korzystania z komputerów w trakcie dalszych studiów (np. w formie komputerowego wspomagania nauczania CAI lub projektowania CAD/CAM) oraz do uczęszczania na inne bardziej zaawansowane zajęcia infomatyczne.

W pełni powszechne zajęcia z elementów informatyki są prowadzone w laboratoriach uczelnianych, wyposażonych w terminale systemu wielodostępnego z dużym komputerem. W przeciwieństwie do laboratorium wyposażonego w pewną liczbę mikrokomputerów lub nawet w sieć komputerową (model taki stał się ostatnio dość popularny w naszym kraju), duży komputer z wielodostępem ułatwia wszystkim użytkownikom korzystanie z oprogramowania systemowego (np. języków, bibliotek programów i innych systemów) oraz umożliwia prowadzenie rozliczeń kont studenckich i kontrolę postępów w nauce.

Liczba szkół w Japonii i stopień ich wyposażenia w komputery (dane z 1983 roku)

Rodzaj szkoły	Liczba wszystkich szkół z podziałem na państwowe i prywatne	Szkoły wyposażone w komputery		Średnia liczba komputerów w szkole
		Liczba	Procent	
Szkoły podstawowe	24802	27	0,1	1,1
	168	9	5,4	2,4
Szkoły średnie	10252	94	0,9	1,3
	550	100	18,2	3,1
Gimnazja	3954	1970	49,8	3,9
	1242	400	32,2	4,8
Uniwersytety	129	–	–	–
	331			

[1] Czytelnika zainteresowanego sorobanem odsyłamy do krótkiej notatki w INFORMATYCE nr 4, 1985, s. 28.

[2] Wszystkie dane liczbowe dotyczące szkolnictwa i skali komputeryzacji szkół w Japonii zostały zaczerpnięte z artykułu „Computerbildung in Japan", Computer Magazin, No. 12, 1986, pp. 38–40.

W Japonii istnieje akademicka sieć komputerowa N-1 łącząca ze sobą duże komputery w siedmiu głównych ośrodkach akademickich (Tokio, Osaka, Sapporo, Sendai, Nagoja, Kioto i Fukuoka). Do sieci tej są dołączone także inne komputery. O jej rozmiarach mogą świadczyć liczby terminali: 1300 działających na liniach dedykowanych i 2600 dołączonych za pośrednictwem publicznej sieci telefonicznej.

CAI – systemy komputerowego wspomagania nauczania

Dane statystyczne zamieszczone w tabeli wskazują bardzo niewielkie nasycenie komputerami szkół pierwszych dwóch typów na początku lat osiemdziesiątych. Nie oznacza to jednak, że nie doceniano dotychczas znaczenia komputerów w szkołach. Badania i prace wdrożeniowe nad wprowadzeniem komputerów do szkół podstawowych w Japonii są bowiem prowadzone od ponad 20 lat, a brak widocznych efektów w samych szkołach należy uznać za pozytywny objaw tej działalności. Po wnikliwej analizie przyjęto bowiem dość oczywiste założenia, iż w szkole podstawowej jest jeszcze zbyt wcześnie, by dzieci uczyć programowania i budowy komputerów, a zastosowanie komputerów do innych celów powinno być umotywowane wzrostem efektywności nauczania. Dalej będzie więc mowa jedynie o komputerowym wspomaganiu nauczania (ang. *Computer Aided Instruction* – CAI). Poza CAI, mikrokomputery są stosowane w szkołach podstawowych do wykonywania eksperymentów i demonstracji obliczeniowych oraz jako elementy składowe innych pomocy dydaktycznych, wzbogacające ich możliwości prezentacji.

Poniższe rozważania są omówieniem doświadczeń zebranych przez zespół Centrum Nauki o Przetwarzaniu Informacji (ang. *Science Information Processing Center*, w skrócie Centrum SIP) Uniwersytetu Tsukuba w Japonii, który od wielu lat zajmuje się projektowaniem sprzętu i oprogramowania dydaktycznego na potrzeby CAI w szkołach podstawowych. Projekty te zrealizowano w postaci laboratorium dydaktycznego pracującego w szkole Takezono Higashi w Ibaraki pod koniec lat siedemdziesiątych[3)].

Przed szczegółowym omówieniem kolejnych etapów rozwoju systemów CAI, warto wspomnieć o trudnościach, jakie musiały być i są pokonywane przez ich twórców i użytkowników. Od samego początku twórcom oprogramowania dydaktycznego w Japonii najwięcej kłopotów sprawiał pisany język japoński, który jest mieszaniną *kanji* (czytaj: *kandżi*), czyli znaków ideograficznych pochodzenia chińskiego i dwóch rodzajów japońskich sylab fonetycznych, zwanych *kana*. Liczba wszystkich kanji nie jest znana. Mówi się o 40 000, chociaż wydany w 1956 roku plakat zawiera ich 50 294! Większość z nich ma jedynie wartość historyczną, a tylko około 5000 znajduje się w użyciu; jednak i to jest zbyt obszernym alfabetem dla bardzo wielu Japończyków. Dlatego w 1947 r. Japońskie Ministerstwo Szkolnictwa opublikowało listę 1 850 tzw. *tōyō* (czytaj: *toojoo)* kanji z zaleceniem, by tylko tych uczyć w szkołach i stosować na co dzień (m. in. w gazetach i w innych środkach masowego przekazu). Obecnie, ten podstawowy zbiór kanji składa się z 1945 znaków tworzących zbiór zwany *jōyō* (czytaj: *dżioojoo)*. Zbiór sylab fonetycznych kana tworzą *katakana* i *hiragana*, każdy złożony z 46 podstawowych znaków. Znaków kany używa się do zapisywania wymowy, znaków kanji, w końcówkach fleksyjnych i łącznikach zdań, do zapisywania słów, które nie mają pisowni za pomocą kanji, oraz do zapisywania słów w ich oryginalnym brzmieniu (rys. 1).

Od samego początku prac nad systemami CAI dla potrzeb japońskiego szkolnictwa podstawowego było oczywiste, że komunikacja między komputerem a uczniami może odbywać się tylko po japońsku i to za pomocą znaków kanji i kany. Dopiero jednak w 1978 r., gdy dostępna stała się odpowiednia technologia wyświetlania znaków, opracowano i przyjęto wzorcowe kody podstawowych znaków języka japońskiego.

W tym samym roku, w Centrum SIP wdrożono procesor tekstów języka japońskiego dla potrzeb programów edukacyjnych. Przed pojawieniem się tego typu procesorów, każdy terminal CAI był wyposażony w projektor przeźroczy lub mikrofilmów, z którego w dowolnej chwili mógł być wyświetlany na monitorze odpowiedni tekst. Było to jedyne, jednak bardzo kosztowne, rozwiązanie.

Innym poważnym utrudnieniem we wprowadzaniu systemów CAI do pierwszych klas szkół podstawowych była i jest prawnie usankcjonowana zasada w japońskim systemie oświatowym, zgodnie z którą dana klasa musi mieć wszystkie przedmioty z tym samym nauczycielem i w ciągu pierwszych sześciu lat nauki nauczyciel może zmienić się najwyżej dwukrotnie. Wprowadzenie do szkół zajęć ze wspomaganiem komputerowym nakłada więc na nauczycieli dodatkowy obowiązek zaznajamiania się z systemami CAI.

Jeszcze innym zmartwieniem twórców tych systemów jest nadal mała efektywność tego sposobu nauczania w porównaniu z tradycyjnymi metodami. Podważa to celowość upowszechniania komputerowo wspomaganego nauczania, zwłaszcza, że nadal jest zbyt mało systemów CAI i niewiele osób pracuje nad ich rozwojem. I wreszcie, biorąc pod uwagę zakładaną skalę rozpowszechniania, koszty całego przedsięwzięcia są za wysokie jak na możliwości oświaty.

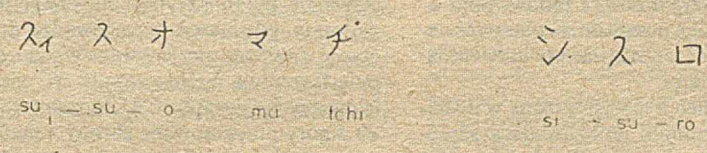
Prace nad systemami CAI w Japonii mają już ponad dwudziestoletnią historię. Pierwsze projekty takich systemów opracowano w latach 1963–1967, a pierwszy eksperymentalny system CAI z kompletnym wyposażeniem sprzętowym i oprogramowaniem zainstalowano w 1971 r. w Japońskim Towarzystwie Popierania Przemysłu Maszynowego. W latach 1974–1976 zainstalowano pierwsze dwa systemy w szkołach w Tokio. Lata 1977–1979 można uznać za okres dojrzałości badań nad systemami CAI. Zbudowano wtedy m. in. system CAI z 17 terminalami w Uniwersytecie Tsukuba, a zespół z Centrum SIP tego uniwersytetu zainstalował pierwszy mikrokomputerowy system CAI w szkole podstawowej Takezono Higashi w Ibaraki. Ważnym osiągnięciem tego okresu było zakończenie prac nad procesorami tekstów języka japońskiego, chociaż niedostępna była jeszcze kolorowa grafika. Lata osiemdziesiąte zapoczątkowały nowy i najbardziej dojrzały okres rozwoju systemów CAI.

Rodzaje mikrokomputerowych systemów CAI

Mikrokomputerowe systemy CAI można klasyfikować w zależności od struktury i rodzaju sterowania w następujący sposób:
- centralnie sterowane z ciągłym podglądem,
- samodzielne,
- samodzielne z ciągłym podglądem.

Systemy pierwszego rodzaju istniały do końca lat siedemdziesiątych, przed erą mikrokomputerów, gdy ze względu na bardzo wysokie koszty CPU sterowanie i przetwarzanie było scentralizowane w jednym dużym komputerze. Wynikiem rewolucji mikrokomputerowej z przełomu lat siedemdziesiątych i osiemdziesiątych było pojawienie się tzw. samodzielnych systemów CAI, czyli zaprojektowanych dla jednego mikrokomputera. Systemy takie są łatwiejsze w projektowaniu i budowie. Oprogramowanie dydaktyczne dla mikrokomputerów jest obecnie oferowane przez wiele firm; tworzą je także sami nauczyciele. Duża część tego oprogramowania jest jednak negatywnie oceniana przez specjalistów. Ze względu na łatwość w tworzeniu samodzielnych systemów CAI, stanowią one większość i ocenia się, iż będą jeszcze przez jakiś czas dominować.

Samodzielne systemy CAI są nieodpowiednie w wypadku stosowania ich do prowadzenia regularnych zajęć z całymi klasami. Jedną z funkcji systemu CAI w tym wypadku powinno być bowiem rejestrowanie wyników nauczania każdego uczącego się, tak aby można



Nazwisko (dwukrotnie) i imię autora artykułu (w języku japońskim nazwisko jest pisane przed imieniem) zapisane fonetycznie za pomocą sylab katakany. Pierwsza wersja nazwiska jest zapisem oryginalnego brzmienia słowa „syslo", zaś druga – odpowiada brzmieniu słowa „syslo" w języku angielskim

[3)] Autor korzystał z informacji zawartych w artykule *„Microcomputer based classroom CAI"* (J. Sci. Educ. Japan B, 1984, 113–122), zaprezentowanej na Trzecim Brytyjsko-Japońskim seminarium poświęconym Wiedzy o Nauczaniu przez profesora Kazuhiko Nakayamę, dyrektora Centrum SIP. Autor dziękuje profesorowi Kazuo Murocie za zorganizowanie odwiedzin w Centrum oraz Pani Akai i Panu Higashibarze za oprowadzenie po laboratorium w Szkole Takezono Higashi i udzielenie wyczerpujących informacji o jego funkcjonowaniu.

było kierować go na najodpowiedniejszą dla niego drogę. Jeśli jedna jednostka tematyczna w systemie CAI jest realizowana w trakcie kilku lekcji, to odpowiedzi uczniów powinny być pamiętane. W tym celu, samodzielne jednostki CAI musiałyby być wyposażone w stacje dysków elastycznych, na których byłby rejestrowany przebieg sesji indywidualnego ucznia. Nauczyciel mógłby wtedy kontrolować postępy uczniów podglądając na bieżąco ich monitory albo przeglądając informacje zarejestrowane na dyskietkach. Przekracza to jednak fizyczne możliwości nauczyciela, nawet w wypadku średnio licznej klasy. Samodzielne systemy CAI są więc najczęściej używane w zajęciach pozalekcyjnych, uzupełniających regularne lekcje. Są wtedy stosowane do testowego nauczania lub sprawdzania wiadomości, czyli w zajęciach, których jedynym wynikiem interesującym nauczyciela jest lista osiągniętych ocen.

Najodpowiedniejszy dla regularnych zajęć w klasach z dużą grupą uczniów jest system CAI trzeciego rodzaju, złożony z samodzielnych stanowisk mikrokomputerowych, po jednym dla każdego ucznia, ze stałym i bezpośrednim dostępem do wszystkich jednostek indywidualnych z wydzielonego stanowiska nauczyciela, w którym ponadto są przechowywane pliki poszczególnych uczniów. W wypadku takiej organizacji systemów CAI, nauczanie jest prowadzone przez indywidualne mikrokomputery, które po zakończeniu sesji przekazują plik z zapamiętanym przebiegiem sesji do komputera nauczyciela. Ocena przebiegu jednej sesji ma wpływ na odpowiednie przygotowanie następnych sesji dla poszczególnych uczniów przez system, który umożliwia nauczycielowi także kontrolę uczniów bezpośrednio w trakcie zajęć. Programy kierujące pracą uczniów rezydują bądź w indywidualnych mikrokomputerach, bądź w centralnym – zależy to od szybkości i zdolności transferu danych między mikrokomputerami. Najodpowiedniejszym więc rozwiązaniem technicznym do prowadzenia CAI w formie regularnych zajęć w licznych klasach jest lokalna sieć mikrokomputerowa w konfiguracji gwiaździstej.

Mikrokomputerowy system CAI w klasie

W 1977 r. Centrum SIP Uniwersytetu Tsukuby utworzyło Laboratorium CAI w szkole podstawowej Takezono Higashi w Ibaraki, by prowadzić badania nad praktyczną stroną możliwości tkwiących w systemach CAI. Na początku przyjęto upraszczające założenie, że cała klasa uczy się w sposób jednorodny. Poza tym, tworząc koncepcję laboratorium uwzględniono następujące wymagania wynikające przede wszystkim z jego usytuowania w szkole podstawowej.

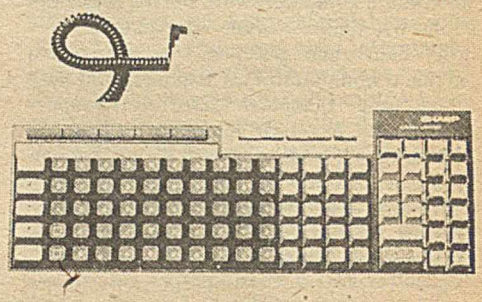
- System powinien być przyjazny dla dzieci, a praca z nim winna zachęcać do nauki i dostarczać im przyjemności i zadowolenia.
- System powinien uwzględniać obecność nauczyciela w klasie i umożliwiać mu indywidualną pracę z uczniami.
- System powinien być wyposażony w dostateczną liczbę terminali umożliwiającą samodzielną pracę uczniów przy terminalach, a w trakcie jednej sesji – pracę wszystkich uczniów w zakresie tej samej jednostki tematycznej.
- System ma umożliwiać uczenie się przez spontaniczną aktywność uczniów zamiast biernego odbierania wiadomości.
- System powinien umożliwiać uczenie się w niewielkich grupach dzieci wymieniających między sobą pomysły i doświadczenia. Uczenie się powinno być związane raczej z pracą zespołową niż z suchym dialogiem dziecka i maszyny.
- Zawartość merytoryczna, struktura i sposób prezentacji przekazywanego materiału nie powinny dopuszczać do powstawania większych różnic między uczniami w stopniu jego opanowania. Może to być osiągnięte na przykład przez opracowanie różnych wersji tego samego tematu: podstawowej, standardowej i zaawansowanej, tak by każdy uczeń mógł wybrać najbardziej odpowiednią dla siebie.
- Korzystanie z możliwości dydaktycznych systemu powinno być łatwe dla przeciętnego nauczyciela, który nie zna programowania komputerów, a jedynie potrafi systematycznie myśleć o przerabianym materiale i umie przewidzieć możliwe reakcje uczniów.
- Nauczyciel powinien mieć możliwość nauczenia się obsługi systemu w ciągu godziny; podobnie jedna jednostka lekcyjna powinna wystarczyć na opanowanie przez uczniów obsługi terminali.
- System powinien być niedrogi i z upływem czasu powinien tanieć.

Pedagogika przywiązuje duże znaczenie do uczenia się w trakcie stawiania pytań i rozstrzygania wątpliwości. Wprowadzając mikrokomputery do klas należy więc położyć duży nacisk na realizację tego procesu, a jednocześnie zadbać o rozbudzenie aktywności uczniów. Niestety, większość systemów CAI jest nastawionych tylko na dostarczanie uczącym się wiadomości. Charakterystyczne cechy omawianego laboratorium CAI odróżniają nieco ten system od innych systemów CAI, w których dla przykładu nie uwzględnia się obecności nauczyciela podczas zajęć, gdyż całym procesem uczenia się kieruje komputer, a uczniowie mogą przerabiać materiał w indywidualnie dobieranym zakresie i szybkości. Zaleceniem japońskiego resortu oświaty jest, by dzieci uczyły się pewnej porcji materiału (ustalonej przez Ministerstwo) w jednorodnie instruowanych grupach. Nie wyklucza to indywidualnego traktowania uczniów. Nauczyciel w laboratorium pomaga osobiście tym uczniom, którzy zgłaszają taką potrzebę najczęściej przez podniesienie ręki. Może także posłużyć się swoim terminalem, by dowiedzieć się, którzy uczniowie potrzebują pomocy, gdyż nie robią dostatecznie szybkich postępów w nauce. W rzeczywistości, nauczyciel porusza się po laboratorium więcej niż podczas innych zajęć, zatem istnieje większa możliwość indywidualizacji nauczania.

Specjaliści uważają, iż uczenie się przez stawianie pytań przynosi najwięcej pożytku wówczas, gdy uczeń przeprowadza doświadczenie, analizuje otrzymywane wyniki, a następnie formułuje ogólną zasadę. W większości istniejących systemów CAI, komputer wyświetla z góry przygotowane ekrany z materiałem uczącym, który najczęściej jest przyjmowany pasywnie przez uczącego się. W wypadku omawianego systemu, komputer kieruje wykonywaniem doświadczenia (którym jest np. nauka ważenia), zebrane wyniki eksperymentów uczniowie omawiają i analizują (najczęściej w grupach), a wnioski wpisują do swoich zeszytów przed przejściem do kolejnego etapu uczenia się (system nie jest wyposażony w żadną drukarkę!). Należy jeszcze dodać, że omawiany system CAI nie służy ani do oceny postępów w nauce, ani do egzaminowania uczniów ze zdobywanych wiadomości. Systemy egzaminacyjne są o wiele prostsze w budowie niż omawiany system CAI.

Wyposażenie i oprogramowanie laboratorium CAI

Każde stanowisko w laboratorium jest wyposażone w mikrokomputer X1 firmy SHARP, zawierający mikroprocesor Z80 i pamięć operacyjną o pojemności 128 KB, stację dysków elastycznych i kolorowy monitor. Dodatkowe 8 KB pamięci RAM jest zarezerwowane dla potrzeb procesora tekstowego 256 znaków kanji. Koszt jednego zestawu wynosi około 300 tys. yenów (w przybliżeniu 2 tys. dolarów) bez oprogramowania i 500 tys. yenów wliczając koszt oprogramowania (dla porównania, 300 tys. yenów jest średnią miesięczną pensją nauczyciela w Japonii).



Fot. 1. Klawiatura japońskiego mikrokomputera szkolnego SHARP X1

Na fot. 1 przedstawiono klawiaturę terminala, na której są umieszczone klawisze ze znakami kany, cyframi i znakami działań oraz pewna liczba klawiszy funkcyjnych, których przeznaczenie zależy od rodzaju zajęć. Jak widać, klawiatura nie zawiera przycisków dla żadnego z 256 znaków kanji używanych przez system. Aby wyświetlić znak kanji należy podać jego brzmienie używając do tego celu znaków kany. Brzmienie nie indentyfikuje jednoznacznie znaku, dlatego są wyświetlane wszystkie kanji o podanym brzmieniu w kolejności częstości stosowania, a użytkownik przy terminalu wybiera poszukiwany przez siebie. W bardziej rozbudowanych procesorach tekstów japońskich można poszukać znaków kanji podając oprócz lub zamiast brzmienia różne inne informacje identyfikujące znak.

Laboratorium składa się z dwóch bardzo starannie zaprojektowanych i urządzonych sal. W mniejszej znajduje się terminal i mikrokomputer nauczyciela wraz z dodatkowym wyposażeniem sieciowym. W dużej sali na 11 stołach są rozmieszczone 44 terminale uczniów. Na każdym, specjalnie zaprojektowanym stole stoją cztery monitory i cztery klawiatury, po dwa zestawy w kolorze białym i czerwonym. Stacje dysków elastycznych znaj-

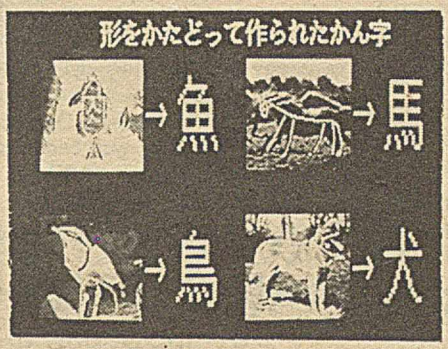
dują się pod stołem. Dzięki specjalnej budowie klawiatury i monitora oraz ukryciu stacji dysków, uczeń nie ma prawie żadnej możliwości manipulowania sprzętem. Umieszczenie kilku terminali na jednym (okrągłym) stole ułatwia formowanie się uczniów w grupy. Podłoga w laboratorium jest pokryta żółtopomarańczową wykładziną, dzięki czemu osiąga się ciepłą atmosferę w sali i między uczniami. Uczniowie pozostawiają przed wejściem swoje obuwie, w którym chodzą po szkole, a w sali mogą siedzieć i pracować także na podłodze. To wszystko po to, by uczeń czuł się swobodniej (Japończycy większość czasu w domach spędzają bez żadnego obuwia siedząc na matach tatami) i nie poczuł najmniejszego „chłodu", towarzyszącego zwykle kontaktom człowieka z maszyną.

Jedną z ważniejszych cech oprogramowania w omawianym systemie CAI jest duże znaczenie, jakie autorzy przywiązali do tzw. wiedzy o wynikach (ang. *knowledge about results*), czyli informacji o odpowiedziach uczniów przekazywanych im bezpośrednio po uzyskaniu odpowiedzi. Dodatkowa zachęta lub stymulacja działania uczącego się mają najczęściej pozytywny wpływ na jego aktywność. Obecnie system dysponuje 400 rodzajami informacji o odpowiedziach zapamiętanych w komputerze. Są to informacje bardzo zbliżone do stosowanych przez nauczycieli w bezpośrednich kontaktach z uczniami. Dużą grupę wyrażeń stanowią pochwały, które zwielokrotniają radość uczących się z powodu dobrze wykonanych poleceń.

Reakcje dzieci na lekcje w laboratorium są bardzo pozytywne. Wiele z nich woli je niż zajęcia prowadzone tradycyjnymi metodami. Przede wszystkim, uczeń ma możliwość przerabiania materiału we własnym tempie. System przetwarza polecenia i dodatkowo pomaga uczniowi, gdy ten gubi się. Uczniowie mogą pomagać sobie wzajemnie, a w ostateczności jest jeszcze nauczyciel. Uczący się bardzo chętnie korzystają z pomocy nauczyciela, jeśli jest to konieczne lub zalecane przez komputer. Bardzo chwalą sobie system nagradzania za dobre odpowiedzi – pochwały zachęcają do uczenia się czegoś więcej.

W ciągu sześciu lat istnienia laboratorium w szkole Takezono Higashi, w Centrum SIP opracowano programy dla 27 jednostek tematycznych, z których każda jest przeznaczona na 5–6 lekcji. Wśród nich 14 dotyczy arytmetyki (42 godziny lekcyjne), 9 – przyrody (46 godzin) i 4 – języka japońskiego (6 godzin).

Zajęcia arytmetyki wspomagane komputerowo mają na celu m. in. wyrobienie wprawy w posługiwaniu się arabskimi cyframi oraz zapisem podstawowych działań arytmetycznych. Należy tutaj pamiętać, że cyfry w niematematycznych tekstach japońskich są zapisywane za pomocą znaków kanji, a w tekstach matematycznych najczęściej są używane cyfry arabskie. Ponadto napisy matematyczne w języku japońskim mają pewne dodatkowe cechy, nieobecne w innych językach. Można podać tutaj dwie ciekawostki z tego zakresu. Pierwszy przykład świadczy o tym, że polska notacja wyrażeń zaproponowana przez polskiego logika J. Łukasiewicza ma swój pierwowzór w... języku japońskim. Otóż wyrażenie arytmetyczne może być poprawnie przeczytane po japońsku tak, jakby było ono zapisane w polskiej notacji. Na przykład, $2 + 2 = 4$ czytą się *„ni to ni tatsu yon ni narimasu"*, co znaczy *„dwa i dwa plus cztery się staje"*. Zatem wyrażenie to jest czytane tak, jakby zapisane było w postaci $2, 2 + 4 =$. Drugi przykład dotyczy sposobu pisania i czytania ułamków i liczb mieszanych. Na przykład, liczbę $2\frac{2}{3}$ czyta się *„ni to san bunno ni"*, co znaczy *„dwa i z trzech [części] dwie"*. Zatem najpierw jest czytany mianownik ułamka, a później licznik. Można to bardzo prosto uzasadnić, powinniśmy bowiem najpierw określić na ile części dzielimy całość, a później ile z nich bierzemy. Co więcej, Japończyk zaczyna pisać ułamek od kreski ułamkowej, potem pisze mianownik, a na końcu licznik. Ta kolejność jest ściśle przestrzegana, gdyż podobnie jak z pisaniem kanji, po ostatecznym efekcie dość łatwo jest poznać w jakiej kolejności były rysowane poszczególne części znaku.



Fot. 2. Wyjaśnienie genezy powstania znaków kanji w trakcie lekcji języka japońskiego

W nauczaniu języka ojczystego, komputerowo wspomagane są zajęcia z przyswajania znaków kanji, podczas których jest także wyjaśniana geneza powstania znaku i są opisywane podstawowe jego cechy jako napisu oraz znaczenia (fot. 2). Lekcje te nie mają na celu nauki pisania, gdyż umiejętność poprawnego pisania znaków kanji zdobywa się na lekcjach kaligrafii prowadzonych tradycyjnymi środkami i metodami.

* * *

W konkluzji, autor omawianego opracowania [3] dzieli się swoimi wątpliwościami, czy systemy CAI mają szansę stać się powszechnym wyposażeniem regularnych zajęć w szkołach podstawowych. Wśród wymienionych wyżej trudności, jedną z najpoważniejszych są olbrzymie (jak na oświatę) koszty instalacji i utrzymania sprzętu w szkołach. Już teraz istnieje jednak duże zapotrzebowanie na systemy CAI w szkołach specjalnych dla dzieci upośledzonych i w szkołach zbiorczych, gdzie garstka nauczycieli nie jest w stanie jednocześnie prowadzić zajęć z wszystkimi uczniami. Systemy CAI opracowane w Centrum SIP pracują w wielu tego typu szkołach w Japonii.

MACIEJ M. SYSŁO
Instytut Informatyki
Uniwersytet Wrocławski

Tryb opracowania norm

Dekada rozwoju norm OSI doprowadziła do stworzenia nowego trybu opracowywania i wprowadzania norm. O ile dotychczasowe normy były oparte na wyrobach o sprawdzonej użyteczności i stabilności, to obecnie powinny one zapewnić przynależność wyrobów do przewidywanych klas funkcjonalności. Takim przewidywanym normom często nie odpowiadają żadne prototypy ani nawet bliskie analogie w istniejących wyrobach lub usługach, które mogłyby służyć jako modele do wyspecyfikowania tych norm. Ze względu na to, że trudno przewidzieć wszystkie zastosowania norm, ich specyfikacje są tak zaprojektowane, by zawierały różne opcje zapewniające przewidywane potrzeby. Stosując taką normę w rzeczywistych systemach należy dokonać wyboru wśród tych opcji.

Pierwszą fazą w cyklu tworzenia nowych norm jest opracowanie modelu odniesienia, który określa lub przewiduje, jakie funkcje ma zapewnić nowa norma. Model ten służy do organizowania i koordynowania powiązań między projektami następnych opracowanych norm. Druga faza to opracowanie samych norm, tak jak to odbywało się w latach sześćdziesiątych i siedemdziesiątych. Trzecia faza polega na sformułowaniu uzgodnień odnośnie opcji, które należy wybrać dla norm wzajemnie zależnych, mających obowiązywać jednocześnie. Uzgodnienia te koordynowane były przez ostatnie pięć lat w USA przez odpowiednią jednostkę Krajowego Urzędu Normalizacji (*National Bureau ot Standards* OSI *Implementors' Workshop*), a w Europie przez Grupę Promocji i Stosowania Norm (*Standards Promotion and Application Group* – SPAG). Mają one być przyjęte przez ISO (Międzynarodowa Organizacja Normalizacyjna) jako Międzynarodowe Zalecenia Normalizacyjne (*International Standarized Profiles* – ISP).

Czwartą fazą jest opracowanie pełnych specyfikacji dla poszczególnych klas zastosowań, opartych na zaleceniach trzeciej fazy. Przykładami tego rodzaju opracowań są projekty MAP, TOP i rządowy protokół OSI (*Government OSI Protocol* – GOSIP). Piątą fazą jest opracowanie i wdrażanie testów zgodności sprawdzających spełnianie wymagań przez wyroby. Fazę tę realizują w USA takie organizacje, jak: Stowarzyszenie Systemów Otwartych (*Corporation for Open Systems* – COS) i Instytut Techniki Przemysłowej (*Industrial Technology Institute* – ITT).

Doświadczenia ubiegłej dekady pokazały, że następuje znaczne zachodzenie na siebie w czasie tych pięciu faz. Każda kolejna faza rozpoczyna się znacznie wcześniej, aniżeli faza poprzednia zaowocuje końcowymi wynikami. W rezultacie obserwuje się duży stopień równoległości tych faz, co wymaga ich koordynacji i uzgadniania różnic w ich przebiegu.

Wzrost sprzedaży sprzętu

Wpływy pięćdziesięciu głównych dostawców sprzętu komputerowego i biurowego wzrosły w pierwszym kwartale 1988 roku o 60,4% w stosunku do roku ubiegłego osiągając wartość 65,93 mld dolarów. Zyski tych przedsiębiorstw wzrosły o 37,9% osiągając 3,27 miliarda. Spowodowało to wzrost inwestycji w tej dziedzinie o 32,9%, choć tempo takie nie da się prawdopodobnie utrzymać w ciągu całego roku. W zakresie oprogramowania i usług wzrost wpływów wyniósł (dla 25 czołowych firm) 18,9% osiągając wartość 3,77 mld dolarów, zyski natomiast utrzymały się na zeszłorocznym poziomie, około 190 mln dolarów. Wpływy 25 głównych przedsiębiorstw w telekomunikacji wzrosły w tym samym okresie o 6% do wartości 47,39 mld dolarów, a zyski wzrosły tu bardziej, bo o 10,9% osiągając 3,87 mld dolarów.

18. konferencja GfI

W październiku ub.r. odbyła się w Hamburgu 18. doroczna konferencja zachodnioniemieckiego towarzystwa informatycznego GfI (Gesellschaft für Informatik). O randze konferencji świadczył fakt uczestnictwa ministra gospodarki RFN Bangemanna. Podczas obrad wypłynęło szereg spraw o kluczowym znaczeniu dla rozwoju informatyki RFN, uznawanej powszechnie za kluczowy element umocnienia pozycji gospodarczej tego kraju w obliczu zagrożeń, jakie niesie rosnąca przewaga technologiczna Japonii i USA.

Sprawy te, mimo że dotyczą kraju znajdującego się na innym niż Polska etapie rozwoju gospodarczego, warto zasygnalizować, gdyż mogą być dla nas pouczające i skłaniające do zastanowienia, a nawet możliwe do bezpośredniego wykorzystania w bieżącej polityce kierunkowania postępu naukowo-technicznego oraz rozwoju społeczno-gospodarczego kraju.

Do najbardziej interesujących informacji z obrad konferencji zaliczyć należy ujawnienie faktu znacznego deficytu kadr informatyków w RFN, ocenianego na 30–40 000 osób. Jednocześnie ostrej krytyce poddano istniejący system kształcenia tych kadr, który nie stwarza właściwych warunków do szybkiej likwidacji tego deficytu. Skrytykowano również obowiązujący system subwencjonowania badań naukowych, który nie uruchomiając mechanizmów konkurencji spowodował, że RFN w żadnej ze specjalizacji informatycznych nie może wykazać się osiągnięciami na szczytowym poziomie światowym.

Z treści wygłoszonych referatów i towarzyszącej im dyskusji ukształtować było można również obraz aktualnych światowych tendencji rozwoju informatyki. Tendencje te można w najbardziej syntetyczny sposób ująć następująco:

1. W dziedzinie sprzętu dominują jeszcze układy 256 Kb, ale coraz bliższy jest okres szerokiego stosowania układów 1 Mb, na które powstało już duże zapotrzebowanie. Zapotrzebowanie to jest jednak w chwili obecnej limitowane przez niedostateczną podaż, wynikającą z ograniczonej jeszcze liczby producentów tych układów. W najbliższym czasie należy natomiast spodziewać się uruchomienia produkcji przemysłowej układów 4 Mb. Przewiduje się również, że proces scalania będzie postępować coraz szybciej i już w 1995 r. można spodziewać się dostaw układów 64 Mb. Ponieważ koszt uruchomienia linii produkcyjnej układów o bardzo dużym stopniu scalenia wynosi obecnie ok. 30 mln dolarów, to przy obowiązującym w tej dziedzinie trzyletnim okresie amortyzacji nakładów inwestycyjnych oraz istniejącym poziomie cen i rentowności sprzedaży układów tego typu, niezbędne jest założenie produkcji na poziomie co najmniej 1 mln układów rocznie, z jednej takiej linii.
2. Przemysł oprogramowania RFN oferuje już liczne rozwiązania rozproszonych systemów baz danych (*Distributed Database Systems*), których sprzedaż natrafia jednak na duże trudności. Wynika to z braku odczuwalnej redukcji kosztów eksploatacji tego rodzaju systemów w porównaniu do dotychczas stosowanych rozwiązań systemów scentralizowanych.
3. W wielu zastosowaniach osiągnięto już podstawowy cel zalecanego modelu OSI (*Open System Interconnection*), który zapewnia komunikację heterogenicznych systemów komputerowych z kompatybilnymi implementacjami standardowych funkcji komunikacyjnych. Obserwować można również dalszą ewolucję modelu OSI w kierunku architektury otwartego przetwarzania rozproszonego (ODP – *Open Distributed Processing*).
4. W skali międzynarodowej powstaje nowa gałąź informatyki – neuroinformatyka. W RFN rozpoczęto w tej dziedzinie intensywne badania naukowe, zarówno w szkołach wyższych, jak i w przemyśle. Polegają one na opracowaniu sprzętu, niezależnego od oprogramowania i wzorowanego na żywych organizmach. Obszar zastosowań takiego sprzętu jest bardzo duży. W USA istnieją już pierwsze doświadczenia z sieciami neuronowymi, które testowano w zastosowaniu do akustycznej kontroli jakości silników oraz stwierdzania wiarygodności kredytobiorców w bankach (!). Oprócz zastosowań praktycznych, należy spodziewać się wpływu tych badań na rozwój wiedzy o biologicznym przetwarzaniu informacji. **(WK)**

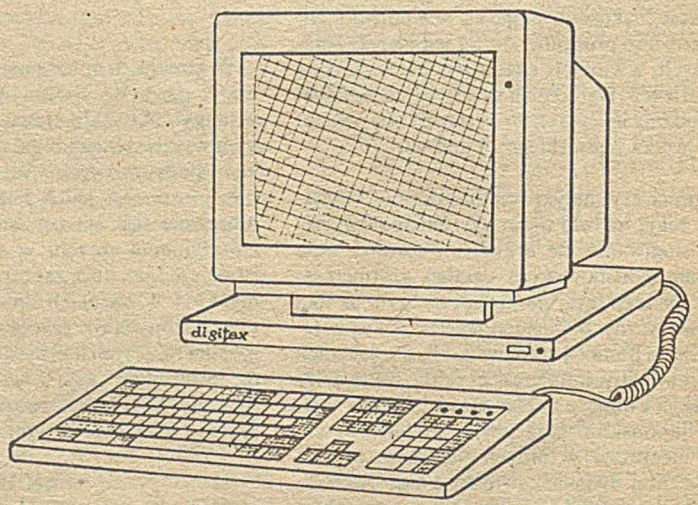
digitex

ZAKŁAD SYSTEMÓW CYFROWYCH
81-832 Sopot, ul. Mickiewicza 20
tel. 51-28-27, tlx 512290 dgtex pl

KOLOROWY TERMINAL GRAFICZNY DXT–125

- emuluje protokół terminali graficznych DEC*– a VT – 125*
- emuluje protokół terminali alfanumerycznych DEC-a VT–100*, VT–52*
- przeznaczony do pracy w konfiguracjach wielodostępnych z komputerami serii SM, PDP11, PC XT/AT pod kontrolą systemów operacyjnych **XENIX, UNIX, RSX, MULTILINK** i innych
- wyświetla informacje nie tylko w postaci znaków alfanumerycznych, ale również w postaci wykresów, rysunków itp., zgodnie z protokołem graficznym **ReGIS***, który m. in. zapewnia kreślenie prostych, okręgów, elips, krzywych, zapełnianie, „dostęp" do każdego punktu itp.

- rozdzielczość grafiki:
  512 × 256 punktów – DXT–125A
  640 × 400 " – DXT–125B
  640 × 480 " – DXT–125C
  800 × 480 " – DXT–125D
- atrybuty: 16 kolorów każdego punktu (lub 8 kolorów i migotanie)
- tryb alfanumeryczny: maks. 60 wierszy po 100 znaków
- współpracuje z dowolną drukarką (złącze równoległe lub szeregowe)
- klawiatura typu PC–XT
- monitor kolorowy 14" (RGB standard, EGA lub Multisync – zależnie od rozdzielczości terminala)



*DEC, VT, ReGIS są zastrzeżonymi znakami firmowymi Digital Equipment Co.

EO/471/89

NIE DAJEMY RECEPT – SPRZEDAJEMY NARZĘDZIA

ELEKTRONIKA FILM KOMPUTER

Agencja Usługowo-Handlowa
Spółdzielni Pracy UNICUM

ul. Barska 3/20, 02–315 Warszawa
tel. 23-67-57, teleks 816955 efcom pl

BIURO HANDLOWE: ul. Dzika 4
Warszawa
tel. 635-23-36, 635-23-37
teleks 816075 medi pl

OFERUJE USŁUGI W DZIEDZINIE KOMPUTERYZACJI PRZEDSIĘBIORSTW

PODEJMUJE SIĘ KOMPLEKSOWEJ OBSŁUGI KONTRAHENTÓW:

- sprzedaż sprzętu mikrokomputerowego (kompletacja, dostawa, serwis gwarancyjny i pogwarancyjny)
- opracowanie oprogramowania użytkowego (wdrożenie, szkolenia)
- instalacja systemów wielodostępnych (SCO XENIX 286, 386) oraz systemów sieciowych (Novell) – licencyjnych

Służymy Państwu:

- doradztwem organizacyjnym
- projektowaniem, oprogramowaniem oraz wdrażaniem systemów dedykowanych konkretnemu użytkownikowi
- projektowaniem unikalnych programów wraz z nadzorem autorskim.

EO/1299/89

Bazy danych

C. N. Prague, J.E. Hammitt: The dBase III Programming Handbook. John Wiley, New York, 1986

W tej książce, autorzy bazując na ideach i aplikacjach prezentowanych w ich pierwszym bestselerowym przewodniku po dBase III, koncentrują swą uwagę na funkcjach systemu dBase III, które w standardowych podręcznikach są zaledwie wzmiankowane. Dają ponadto wgląd w różnorodne możliwości zastosowań, których zapewne większości użytkowników nie udałoby się odkryć samodzielnie.

Aspekty techniczne są potraktowane w sposób łatwy do śledzenia, głównie z tego względu, że autorzy prezentują także szczegóły sprzętowe, związane z użytkowaniem tego mocnego systemu zarządzania bazą danych. Pokazują również, jak unikać pułapek wynikających z relatywnie małej szybkości przetwarzania systemu dBase III. W tym celu proponują przetwarzanie z wykorzystaniem, gdzie tylko możliwe, metod dostępu losowego; tworzenie ekstraktów (wyciągów) z plików przed drukowaniem raportów; oddzielanie od siebie tych operacji, które w wypadku przetwarzania ich razem zajmują dodatkową ilość czasu; zestawianie programów wraz z procedurami w celu przyspieszenia interpretacji.

W książce przedstawiono także, oprócz wielu innych kwestii:
– użycie kompilatora Runtime firmy Ashton-Tate, programy dla dyskowej pamięci podręcznej (ang. *cache*), spoolery do drukowania oraz inne udogodnienia PC-DOS-a jako alternatywne środki dla zwiększania szybkości działania dBase III,
– zaawansowane techniki do obsługi plików i ich odtwarzania (ang. *recovery*),
– użycie dBase III jako niezależnego języka programowania,
– możliwości wyjścia poza dBase III w celu wyeksportowania lub zaimportowania danych z innych źródeł oraz uzupełnienie systemu dBase w postaci samouczka (funkcji HELP).

C. N. Prague, J. E. Hammitt: Programming with dBase III Plus. John Wiley, New York, 1986

Autorzy naświetlają usprawnienia w nowej generacji systemu dBase, które czynią dBase III Plus jednym z najbardziej wyrafinowanych i wszechstronnych pakietów programowych dostępnych na dzisiejszym rynku. Wśród tematów omówionych w książce znajdują się, między innymi: ponad 50 nowych poleceń i funkcji; nowe udogodnienia, takie jak generator aplikacji (tj. programów użytkowych) i zaawansowany system zapytań; dziesięciokrotnie szybsze metody sortowania i zdolności indeksowania niż w dBase III; rozszerzanie wielkości plików.
Spis treści. Część I. Podstawy programowania; Opracowanie programów; Odwzorowanie procesów; Działanie języka; Kodowanie, testowanie i usuwanie błędów. Aspekty związane z prowadzeniem interesów (przedsiębiorstwa); Generatory plansz, raportów i aplikacji. **Część II.** Podstawy baz danych. Co to jest baza danych?; Tworzenie i użytkowanie struktury bazy danych; Rekordy bazy danych; Sortowanie i indeksowanie; Manipulowanie i wyszukiwanie w bazie danych; Relacje danych. **Część III.** Programowanie w dBase III Plus. Tworzenie zastosowań w systemie dBase III Plus; Polecenia systemu dBase III Plus; Kodowanie menu dla selekcji; Kodowanie plansz do wprowadzania danych; Przystosowanie i drukowanie raportów; Dodawanie rekordów w systemie dwuplikowym; Produkowanie raportów w systemie wieloplikowym.

M. A. Sherif: Database Projects. A Framevork for Effective Management. John Wiley, New York, 1988

Przez ostatnie dziesięć lat, nowoczesne metody analizy ustrukturyzowanych systemów zmierzały do proponowania narzędzi i technik do analizy danych, modelowania danych i przygotowania architektur systemów sterowanych danymi. Książka ta łączy zainteresowania aspektami technologicznymi z przeglądem krytycznych zagadnień organizacyjnych oraz wymagań pojawiających się wówczas, gdy dane są centralnie koordynowane i administrowane. Stawia wyzwanie pod adresem konwencjonalnego poglądu, że dominującym problemem w implementacji bazy danych jest niechęć użytkujących ją biur do współdzielenia danych.

Autor, używając podejścia empirycznego, bazującego na jego własnym doświadczeniu i na analizie rzeczywistych projektów współdzielenia danych oraz przypadków studialnych, omawia pojęcia i terminy opisujące polityczne aspekty projektu współdzielonej bazy danych. Pokazuje, jak cenne są spory i dyskusje między analitykami danych i analitykami systemów aplikacyjnych na temat ich roli i odpowiedzialności, w wypracowywaniu rozwiązań organizacyjnych dla różnych problemów natury politycznej.

Przez całą książkę przewija się idea, że pomyślna implementacja bazy danych wymaga zarówno kompetencji technicznych jak i przygotowania menedżerskiego. Istotnym novum w tej książce w porównaniu do istniejącej literatury (nastawionej w większości na zaawansowanych praktyków i zwykłych użytkowników) jest jasne zdefiniowanie zmian organizacyjnych, które z pewnością muszą wystąpić, jeśli przedsiębiorstwo zdecyduje się zastosować bazodanowe podejście do rozwijania systemów informacyjnych.

S. Spaccapietra (Ed.): Entity-Relationship Approach – Ten Years of Experience in Information Modelling. Proc. 5th International Conference on Entity-Relationship Approach, Dijon, France, 17–19 November 1986. North-Holland, Amsterdam, 1987

Prace opisywane w artykułach prezentowanych na tej konferencji stanowią aktualny przegląd istotnych rezultatów uzyskanych w ciągu dziesięcioletnich doświadczeń w modelowaniu i zarządzaniu informacją z użyciem podejścia Entity-Relationship (byt-związek). Dlatego dają one wgląd w oczekiwane właściwości systemów zarządzania bazami danych następnych generacji.

W zamierzeniu konferencja miała służyć jako punkt kontrolny do oceny użyteczności podejścia byt-związek w procesie projektowania oraz przydatności tego podejścia jako narzędzia operacyjnego. Miała także dać wgląd w teorię modelu byt-związek oraz w perspektywy rozwojowe. Zaprezentowane prace pokrywają wszystkie aspekty zarządzania danymi, od aspektów teoretycznych do zastosowań: dotyczą także nowo powstających dziedzin, takich jak systemy z bazą wiedzy, systemy obiektowe oraz systemy wielonośnikowe. Książka zawiera zarówno artykuły techniczne, jak i materiał wykładowy. Jest przeznaczona dla szerokiego grona odbiorców, od pracowników naukowych do praktyków w dziedzinie zarządzania danymi.
Spis treści (wybór). Słowo wstępne (F. Bodart). Integracja danych a systemy z bazą wiedzy – działalność w projekcie ESPRIT (U. L. Haass). Wiedza a dane – przegląd na marginesie konferencji IFIP DS-2 (R. Meersman). Obiektowe systemy baz danych – raport roboczy (K. R. Dittrich). Modelowanie obiektów złożonych dla wielonośnikowych baz danych (M. E. Adiba). Pozostałe sesje. Specyfikacje wykonywalne. Teoria baz danych. Od konceptualizacji do implementacji. Systemy oparte na zależnościach funkcjonalnych. Systemy zarządzania bazą danych oparte na podejściu Entity-Relationship. Narzędzia do projektowania. Integracja narzędzi. Zastosowania. Doświadczenia użytkowników z podejściem Entity-Relationship. Wykłady.

W. RZECZKOWSKI

Ogłoszenia ● Ogłoszenia ● Ogłoszenia ● Ogłoszenia

Spółdzielnia Rzemieślnicza „ELMECH", ul. Dobra 56, 00–312 Warszawa, oferuje OWIJARKI ELEKTRYCZNE (pistoletowe) do połączeń „wire-wrap", przystosowane do drutu Ø 0,20–0,35 mm. Informacje – telefon 22-94-46.

EO/494/89

Ogłoszenia ● Ogłoszenia ● Ogłoszenia ● Ogłoszenia

Koczkodaj W. W.: Arkusz kalkulacyjny jako kolejny krok w kierunku nieprogramowania

INFORMATYKA 1989, nr 6, s. 1

Omówienie zalet stosowania uniwersalnych pakietów programowych na przykładzie pakietu arkusza kalkulacyjnego Lotus 1–2–3.

Szczerbiński Z.: Nowa generacja superkomputerów Cray (1)

INFORMATYKA 1989, nr 6, s. 6

Pierwsza część charakterystyki nowej generacji superkomputerów Cray, zawierająca omówienie architektury i oprogramowania różnych modeli maszyny Cray X-MP, a także różnic w porównaniu do modelu Cray-1.

Iszkowski W.: Przyspieszanie obliczeń przez ich rozpraszanie (2). Rozpraszanie programu

INFORMATYKA 1989, nr 6, s. 9

Druga część przeglądu obecnie stosowanych metod zwiększania szybkości działania komputerów, zawierająca omówienie rozwiązań programowych.

Skrzymowski J.: Nowa norma Fortranu (2)

INFORMATYKA 1989, nr 6, s. 12

Druga część charakterystyki języka Fortran 88, zawierająca omówienie podprogramów, modułów oraz instrukcji sterujących tego języka.

Bielecki J.: Grafika w Turbo Pascalu 4.0

INFORMATYKA 1989, nr 6, s. 15

Szczegółowa charakterystyka możliwości oraz przykłady programowania grafiki w języku Turbo Pascal 4.0.

Koczkodaj W. W.: The spreadsheet as a successive step towards nonprogramming

INFORMATYKA 1989, No. 6, p. 1

Discussion on advantages of universal program package application at example of Lotus 1-2-3 spreadsheet.

Szczerbiński Z.: New generation of the Cray supercomputer (1)

INFORMATYKA 1989, No. 6, p. 6

First part of characteristics of the Cray supercomputer new generation, which includes discussion of different Cray X-MP models, as well as of differences in comparison with the Cray-1 model.

Iszkowski W.: Acceleration of computations through scattering (2). Scattering of the program

INFORMATYKA 1989, No. 6, p. 9

Second part of survey of the to-day used methods for computer speed increasing, which includes discussion of software solutions.

Skrzymowski J.: New standard of Fortran (2)

INFORMATYKA 1989, No. 6, p. 12

Second part of characteristics of the Fortran 88 language, which includes discussion of the language subprograms, modules and control statements.

Bielecki J.: Graphics in Turbo Pascal 4.0

INFORMATYKA 1989, No. 6, p. 15

Detailed characteristics of possibilities and examples of programming in the Turbo Pascal 4.0 language.

Koczkodaj W. W.: Kalkulationsprogramm als nachfolgender Schritt in Richtung des Nichtprogrammierens

INFORMATYKA 1989, Nr. 6, S. 1

Eine Besprechung von Vorteilen der universellen Softwarepaketeanwendung auf dem Beispiel des Lotus 1-2-3-Kalkulationsprogrammes.

Szczerbiński Z.: Neue Generation von Cray-Supercomputern (1)

INFORMATYKA 1989, Nr. 6, S. 6

Erster Teil einer Charakteristik von neuen Generation des Cray-Supercomputers, der eine Besprechung der Architektur und Software von verschiedenen Cray X-MP Modellen, sowie der Unterschiede im Vergleich mit dem Cray-1-Modell umfasst.

Iszkowski W.: Beschleunigung von Berechnungen durch Zerstreuung (2). Zerstreuung des Programmes

INFORMATYKA 1989, Nr. 6, S. 9

Zweiter Teil eines Übersichts von heutzutage angewendeten Methoden für Computergeschwindigkeitssteigerung, der eine Besprechung von Softwarelösungen umfasst.

Skrzymowski J.: Neue Fortran-Norm (2)

INFORMATYKA 1989, Nr. 6, S. 12

Zweiter Teil einer Charakteristik von Fortran 88-Sprache, der eine Besprechung von Unterprogrammen, Modulen und Steuerungsanweisungen umfasst.

Bielecki J.: Graphik in Turbo Pascal 4.0

INFORMATYKA 1989, Nr. 6, S. 15

Eine detaillierte Charakteristik von Möglichkeiten und Programmierungsbeispiele der Graphik in der Turbo Pascal 4.0-Sprache.

Konferencje

EUROMICRO '89

W dniach 4–8 września br. odbędzie się w Kolonii (RFN) 15. doroczna międzynarodowa konferencja EUROMICRO, której organizatorem jest holenderska instytucja o takiej samej nazwie. Trwałą pozycję tej konferencji w międzynarodowym kalendarzu imprez informatycznych w pełni potwierdza jej 15-letnia kontynuacja, jaką może poszczycić się tylko niewielu organizatorów. Należy przypomnieć, że konferencje EUROMICRO są adresowane w pierwszym rzędzie do tych przedstawicieli środowisk naukowych, gospodarczych i administracji publicznej, którzy są zainteresowani problemami rozwoju technologii i zastosowań mikrokomputerów.

Hasło przewodnie EUROMICRO '89 „Instrumenty projektowania lat dziewięćdziesiątych" jednoznacznie określa punkt ciężkości zainteresowań organizatorów, a tym samym tematyki obrad. Obrady te podzielono na sesje naukowe, obejmujące wygłoszenie zakwalifikowanych referatów i dyskusje oraz sesje komunikatów, obejmujące zwięzłe doniesienia dotyczące głównie aktualnych wyników zastosowań. Oprócz tego, po zeszłorocznym sukcesie postanowiono włączyć do programu konferencji wydzieloną sesję „muzyka komputerowa" obejmującą kompleks zagadnień związanych z tym tematem.

Szczegółowa tematyka konferencji obejmuje pięć głównych, zgodnych ze światowymi trendami rozwoju, obszarów zainteresowań:

- inżynieria oprogramowania i jej wymagania,
- stacje robocze,
- narzędzia i metodologie,
- architektury,
- zaawansowane zastosowania.

O szczegółowe informacje na temat konferencji należy zwracać się pod adres:

EUROMICRO, Attn.: Mrs. Chiquita Snippe – Marlisa
P. O. Box 545
7500 AM Enschede, The Netherlands

PC ARK

PC-ARK
SPÓŁKA Z O.O.

poleca

ul. Jaracza 3
00–378 Warszawa

NOWOCZESNE * PRECYZYJNE
* WAGI ELEKTRONICZNE *
* analityczne * laboratoryjne *
* jubilerskie * przemysłowe *

– dokładność od 0,1 mikrograma do 10 gramów
– zakres od 3 gramów do 300 kilogramów
– możliwość współpracy z drukarkami i systemami komputerowymi

Sieć punktów serwisowych na terenie całego kraju prowadzi:
– instalacje
– okresowe przeglądy konserwacyjne
– usługi gwarancyjne i pogwarancyjne

tel. 26-09-09, 26-27-94, 26-41-18
tlx 8116962 pc pl

EO/1261/88

Słownik pojęć i terminów z dziedziny rozpoznawania i przetwarzania obrazów (2)

PASOWANIE OBRAZÓW *(image matching)* – technika SEGMENTACJI OBRAZU polegająca na wykrywaniu WZORCÓW przez porównywanie dwóch OBRAZÓW tej samej sceny.

PASOWANIE SZABLONÓW *(template matching, matched filtering)* – technika SEGMENTACJI OBRAZU lub ROZPOZNAWANIA OBRAZÓW polegająca na wykrywaniu WZORCÓW i kształtów przez ich porównywanie z zadanymi wzorcami i kształtami, zwanymi szablonami.

POCIENIANIE *(thinning)* – technika PRZETWARZANIA OBRAZÓW, w której OBSZARY są redukowane do zbiorów (cienkich) KRZYWYCH.

PODZIAŁ OBSZARU *(region partitioning)* – technika SEGMENTACJI OBRAZU, w której OBSZARY są tworzone przez wielokrotne dołączanie podobszarów podobnych ze względu na POZIOM JASNOŚCI lub TEKSTURĘ oraz przez wielokrotne odłączanie podobszarów, które są niepodobne.

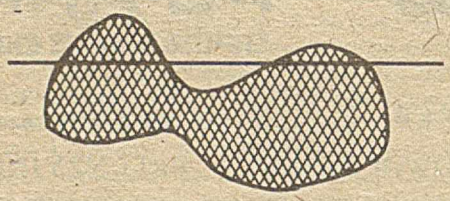
POLE *(area)* – liczba DWELI w podzbiorze OBRAZU.

POZIOM JASNOŚCI, POZIOM SZAROŚCI, ODCIEŃ *(gray level, gray shade, gray tone)* – wartość związana z DWELEM, reprezentująca JASKRAWOŚĆ nieskwantowanej SCENY w SĄSIEDZTWIE punktu OBRAZU reprezentowanego przez DWEL.

PROGOWANIE *(thresholding)* – proces wytwarzania OBRAZU DWÓJKOWEGO z OBRAZU o wielu POZIOMACH JASNOŚCI za pomocą przypisania każdemu wyjściowemu DWELOWI wartości 1, jeżeli POZIOM JASNOŚCI odpowiadającego mu DWELA wejściowego jest nie niższy niż pewna wartość progowa, a wartości 0, jeżeli POZIOM JASNOŚCI DWELA wejściowego jest niższy od tego progu.

PRÓBKOWANIE *(sampling)* – technika PRZETWARZANIA OBRAZU, w której JASKRAWOŚĆ lub kolor OBRAZU mierzy się w dyskretnych punktach tworzących zwykle prostokątną tablicę.

PRZEKRÓJ *(cross-section)* – przecięcie linią prostą OBSZARU lub OBRAZU (rys. 1).



Rys. 1. Ilustracja przekroju

PRZESTRZEŃ CECH *(feature space)* – zbiór n-elementowych krotek $(x_1, x_2, \ldots, x_n)$, używanych do reprezentowania CECH każdego WZORCA, w celu przypisania tych WZORCÓW do określonych KLAS.

PRZESTRZEŃ POMIAROWA *(measurement space)* – zbiór n-elementowych krotek $(x_1, x_2, \ldots, x_n)$ stanowiących wynik n pomiarów każdego analizowanego WZORCA. PRZESTRZEŃ POMIAROWA zawierą surowe dane, z których wydzielane są CECHY w celu późniejszego KLASYFIKOWANIA OBRAZÓW.

PRZETWARZANIE OBRAZU *(image processing, picture processing)* – wykonywanie różnego rodzaju przekształceń na OBRAZIE za pomocą komputera.

PRZETWARZANIE WSTĘPNE *(preprocessing)* – operacje wykonywane przed zasadniczym przetworzeniem, np. w ROZPOZNAWANIU OBRAZÓW upraszczanie WZORCÓW, aby ułatwić ich KLASYFIKOWANIE.

PRZYSTAWANIE *(congruencing)* – technika PRZETWARZANIA OBRAZÓW, w której dwa OBRAZY tej samej SCENY są przekształcane w ten sposób, że rozmiar, kształt, położenie i orientacja wszystkich OBIEKTÓW na obydwu OBRAZACH są takie same.

REJESTROWANIE OBRAZU *(image registration)* – proces umiejscowienia dwóch OBRAZÓW tej samej SCENY względem siebie w ten sposób, że odpowiadające sobie punkty na OBRAZACH reprezentują ten sam punkt na SCENIE.

REKONSTRUOWANIE OBRAZU *(reconstruction, image reconstruction)* – proces odzyskiwania OBRAZU ze składowych jego wektora CECH (np. POZIOMÓW JASNOŚCI), branych wzdłuż pasków lub płatów tego OBRAZU.

ROZCIĄGANIE KONTRASTU *(contrast stretching)* – technika WZMACNIANIA OBRAZU, w której widzialność OBRAZU jest poprawiana przez zwiększenie KONTRASTU między podzbiorami OBRAZU i ich DOPEŁNIENIAMI.

ROZDZIELCZOŚĆ *(resolution)* – stopień odróżnialności blisko rozstawionych OBIEKTÓW na OBRAZIE.

ROZMAZANY *(blurred)* – dotyczący elementów OBRAZU, które są niewyraźne lub nierozróżnialne (przec. OSTRY).

ROZPOZNAWANIE OBRAZÓW, ROZPOZNAWANIE WZORCÓW *(pattern recognition, machine recognition)* – analizowanie, opisywanie IDENTYFIKOWANIE I KLASYFIKOWANIE OBIEKTÓW, zdarzeń lub innych znaczących regularności za pomocą automatycznych lub półautomatycznych środków.

ROZPOZNAWANIE OBRAZÓW PIKTOGRAFICZNYCH *(pictorial pattern recognition)* – ROZPOZNAWANIE OBRAZÓW w danych wizualnych lub na ilustracjach.

ROZPOZNAWANIE OBRAZÓW STATYSTYCZNE *(statistical pattern recognition)* – metoda ROZPOZNAWANIA OBRAZÓW wykorzystująca rachunek prawdopodobieństwa i metody statystyczne do przypisywania WZORCÓW określonym KLASOM.

ROZPOZNAWANIE OBRAZÓW STRUKTURALNE *(structural pattern recognition)* – metoda ROZPOZNAWANIA OBRAZÓW, w której WZORCE są reprezentowane przez pierwotniki i zależności między nimi, w celu opisania i sklasyfikowania struktury WZORCA.

ROZPOZNAWANIE OBRAZÓW SYNTAKTYCZNE *(syntactic pattern recognition)* – rodzaj STRUKTURALNEGO ROZPOZNAWANIA OBRAZÓW, w którym identyfikowane są pierwotniki i zależności między nimi we WZORCACH wyrażonych w naturalnym bądź sztucznym języku.

ROZROST OBSZARU *(region growing)* – technika SEGMENTACJI OBRAZU, w której OBSZARY są tworzone przez wielokrotne dołączanie podobszarów podobnych ze względu na POZIOM JASNOŚCI lub TEKSTURĘ.

SĄSIEDZTWO *(neighborhood)* – zbiór DWELI stykających się z danym DWELEM.

SCENA *(scene)* – wydzielony obszar trójwymiarowej przestrzeni, będący zbiorem TRELI wraz z ich CECHAMI.

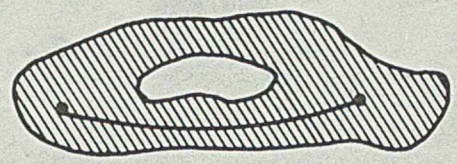
SEGMENTACJA OBRAZU *(segmentation, image segmentation, object extraction)* – proces dzielenia OBRAZU na OBSZARY (podobrazy).

SEGMENTACJA WZORCÓW *(pattern segmentation)* – proces określania, które z OBSZARÓW OBRAZU tworzą WZORZEC

SKLEJANIE KRAWĘDZI *(edge linking)* – technika przetwarzania OBRAZU polegająca na łączeniu DWELI sąsiednich, oznaczonych jako DWELE krawędziowe, tak aby tworzyły KRAWĘDŹ OBSZARU.

SPICZASTOŚĆ *(acutance)* – miara OSTROŚCI KRAWĘDZI na OBRAZIE.

SPÓJNY *(connected)* – dotyczący podzbioru OBRAZU, którego dowolne dwa punkty można połączyć ŁUKIEM całkowicie zawartym w tym podzbiorze (rys. 2).

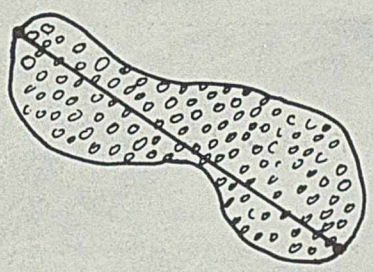


Rys. 2. Ilustracja obszaru spójnego

SZKIELETYZACJA *(skeletization)* – technika odwracalnego POCIENIANIA OBRAZU polegająca na zachowaniu istotnych właściwości jego struktury. Szkielet OBIEKTU jest miejscem geometrycznym środków okręgów wewnętrznych OBIEKTU, ściśle stycznych do jego BRZEGÓW.

SZUM *(noise)* – niezależne dane, przeszkadzające w rozpoznawaniu i interpretowaniu danych użytecznych.

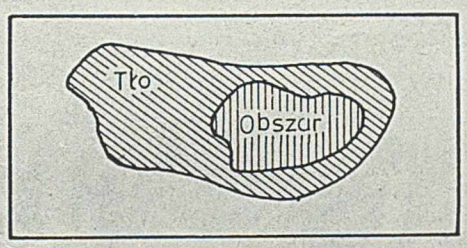
ŚREDNICA PODZBIORU *(diameter)* – maksymalna odległość między dwoma dowolnymi DWELAMI w podzbiorze OBRAZU (rys. 3).



Rys. 3. Ilustracja średnicy

TEKSTURA *(texture)* – atrybut reprezentujący przestrzenne ułożenie POZIOMÓW JASNOŚCI DWELI w OBRAZIE.

TŁO *(background)* – spójne składowe OBRAZU, które leżą wewnątrz DOPEŁNIENIA OBSZARU i otaczają go (rys. 4).

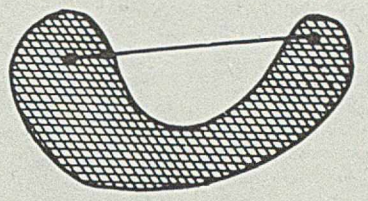


Rys. 4. Ilustracja tła

TREL *(volume element, voxel)* – najmniejszy niepodzielny element trójwymiarowej (dyskretnej) przestrzeni. Termin jest skrótem zwrotu *trójwymiarowy element*.

SKUPIENIE, WIĄZKA *(chluster)* – zbiór DWELI (w PRZETWARZANIU OBRAZÓW) lub zbiór punktów w PRZESTRZENI CECH (w ROZPOZNAWANIU OBRAZÓW) blisko położonych lub podobnych do siebie w pewien sposób.

WKLĘSŁY *(concave)* – odnoszący się do OBSZARU, w którym CECHA geometryczna OBSZARU charakteryzująca się tym, że co najmniej jeden odcinek prostej między dwoma punktami OBSZARU nie leży całkowicie w tym OBSZARZE (rys. 5, przec. WYPUKŁY).



Rys. 5. Ilustracja obszaru wklęsłego

WNĘTRZE *(interior)* – zbiór DWELI OBSZARU w OBRAZIE CYFROWYM, które nie sąsiadują z DOPEŁNIENIEM tego OBSZARU (przec. BRZEG).

WYDZIELANIE CECH *(feature extraction)* – etap ROZPOZNAWANIA OBRAZÓW, w którym na podstawie wyników pomiarów lub obserwacji poszukuje się CECH, które mogą być użyte do przypisywania WZORCÓW do określonych KLAS.

WYGŁADZANIE *(smoothing, noise cleaning, noise suppression)* – technika WZMACNIANIA OBRAZU polegająca na zmniejszaniu wpływu SZUMU w OBRAZIE pierwotnym.

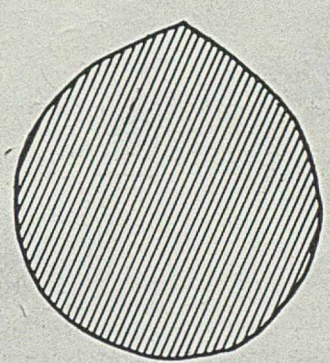
WYKRYWANIE BRZEGÓW, DETEKCJA BRZEGÓW *(border detection, border delineation)* – technika SEGMENTACJI OBRAZU, polegająca na znajdowaniu BRZEGÓW w tym OBRAZIE.

WYKRYWANIE KRAWĘDZI, DETEKCJA KRAWĘDZI *(edge detection)* – technika SEGMENTACJI OBRAZU, polegająca na znajdowaniu DWELI krawędziowych przez sprawdzenie ich SĄSIEDZTWA.

WYKRYWANIE RÓŻNIC, DETEKCJA RÓŻNIC *(change detection)* – technika PRZETWARZANIA OBRAZU, polegająca na porównywaniu odpowiednich DWELI na rejestrowanych OBRAZACH, w celu znalezienia zmian.

WYOSTRZANIE *(sharpening)* – technika WZMACNIANIA OBRAZU polegająca na zmniejszeniu efektu ROZMAZANIA w OBRAZIE pierwotnym.

WYPUKŁY *(convex)* – CECHA geometryczna OBSZARU charakteryzująca się tym, że każdy odcinek prostej między dwoma dowolnymi punktami OBSZARU jest całkowicie zawarty w tym OBSZARZE (rys. 6, przec. WKLĘSŁY).



Rys. 6. Ilustracja obszaru wypukłego

WZMACNIANIE KRAWĘDZI *(edge enhancement)* – technika WZMACNIANIA OBRAZU polegająca na WYOSTRZANIU KRAWĘDZI za pomocą zwiększenia KONTRASTU między POZIOMAMI JASNOŚCI DWELI krawędziowych i niekrawędziowych.

dokończenie na s. 18

Ceny ogłoszeń

Od 1 stycznia 1989 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia
- ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona – 70 tys., 3/4 s. – 60 tys., 2/3 s. – 55 tys., 1/2 s. – 50 tys., 1/3 s. – 45 tys., 1/4 s. – 40 tys., 1/8 s. – 30 tys., poniżej 1/8 s. – 200 zł za cm^2.

Dodatki do ceny podstawowej
- za każdy dodatkowy kolor + 30%,
- za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%,
- za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%,
- za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%,
- za zamieszczenie ogłoszenia na II i III stronie okładki + 50%.

Zniżki
dotyczą ogłoszeń – całkowitych powtórzeń
- za ogłoszenia 3–5-krotne – 10%
- za ogłoszenia 6–10-krotne – 20%
- za ogłoszenia 11-krotne i powyżej – 30%
- za artykuły i wkładki reklamowe wykonane przez zleceniodawcę – 40%
- za biuletyny i bloki reklamowe – 60%

W innych uzasadnionych wypadkach dopuszcza się stosowanie rabatów specjalnych.

Ceny nadbitek reklamowych

wkładka 2 s. A4	
– nakład do 500 egz.	30 tys. zł
– za każde następne 500 egz.	25 tys. zł
wkładka 4 s. A4	
– nakład do 500 egz.	60 tys. zł
– za każde następne 500 egz.	50 tys. zł

Ceny usług nie wymienionych w niniejszym cenniku będą ustalane każdorazowo (w oparciu o kalkulację), jako ceny umowne.

W wypadku rezygnacji Zleceniodawcy z wykonania zamówienia przed przekazaniem materiałów do druku – ponosi on koszty w wysokości 25% wartości zlecenia.
W wypadku rezygnacji – gdy materiały są już w druku – Zleceniodawca ponosi pełne koszty ogłoszenia.

Niniejszy cennik dotyczy wyłącznie ogłoszeń firm krajowych i obowiązuje od 1 stycznia 1989 r. Ogłoszenia przyjęte przed tym terminem będą rozliczane według dotychczas obowiązującego cennika.

Ogłoszenia przyjmowane są przez:
Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-950 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

Warunki prenumeraty w roku 1989

Prenumeratorzy zbiorowi – jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia).

Blankiety te będą dostarczane dotychczasowym prenumeratom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemne lub telefoniczne) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni – osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: Państwowy Bank Kredytowy III/O Warszawa nr 370015-7490-139-11.

Prenumerata ulgowa – przysługuje wyłącznie osobom fizycznym – członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma.

Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumerata ze zleceniem wysyłki za granicę – zamawia się tak, jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy.

Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach:
- do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny,
- do 28 lutego na II, III i IV kwartał oraz II półrocze,
- do 31 maja na III i IV kwartał oraz II półrocze,
- do 31 sierpnia na IV kwartał.

Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela – Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21 wew. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism – można nabyć za gotówkę w Kluhie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 26-80-16), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

miesięczna		kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300 zł	60 zł	900 zł	180 zł	1800 zł	360 zł	3600 zł	720 zł